THE UNIVERSITY *of* EDINBURGH

A Strategy for the Visual Recognition of Objects in an industrial environment

Aravinda S. Athukorala

Ph.D. Thesis

University of Edinburgh

1985

Abstract

This thesis is concerned with the problem of recognizing industrial objects rapidly and flexibly. The system design is based on a general strategy that consists of a generalized local feature detector, an extended learning algorithm and the use of unique structure of the objects. Thus, the system is not designed to be limited to the industrial environment.

The generalized local feature detector uses the gradient image of the scene to provide a feature description that is insensitive to a range of imaging conditions such as object position, and overall light intensity. The feature detector is based on a representative point algorithm which is able to reduce the data content of the image without restricting the allowed object geometry. Thus, a major advantage of the local feature detector is its ability to describe and represent complex object structure. The reliance on local features also allows the system to recognize partially visible objects.

The task of the learning algorithm is to observe the feature description generated by the feature detector in order to select features that are reliable over the range of imaging conditions of interest. Once a set of reliable features is found for each object, the system finds unique relational structure which is later used to recognize the objects. Unique structure is a set of descriptions of unique subparts of the objects of interest. The present implementation is limited to the use of unique local structure. The recognition routine uses these unique descriptions to recognize objects in new images. An important feature of this strategy is the transference of a large amount of processing required for graph matching from the recognition stage to the learning stage, which allows the recognition routine to execute rapidly.

The test results show that the system is able to function with a significant level of insensitivity to operating conditions; The system shows insensitivity to its 3 main assumptions -constant scale, constant lighting, and 2D images- displaying a degree of graceful degradation when the operating conditions degrade. For example, for one set of test objects, the recognition threshold was reached when the absolute light level was reduced by 70%-80%, or the object scale was reduced by 30%-40%, or the object was tilted away from the learned 2D plane by 30°-40°. This demonstrates a very important feature of the learning strategy: It shows that the generalizations made by the system are not only valid within the domain of the sampled set of images, but extend outside this domain. The test results also show that the recognition routine is able to execute rapidly, requiring 10ms-500ms (on a PDP11/24 minicomputer) in the special case when ideal operating conditions are guaranteed. (Note: This does not include pre-processing time).

This thesis describes the strategy, the architecture and the implementation of the vision system in detail, and gives detailed test results. A proposal for extending the system to scale independent 3D object recognition is also given.

Acknowledgements

My thanks are due to many people for their help and encouragement during the time spent on this work. I would firstly like to thank my supervisors Robin Popplestone and Andrew Wallace for their comments on drafts of this thesis, James McGinley for very useful discussions, Jim Howe, Richard Tobin and Ken Gourlay for their comments on parts of this thesis, and the Napier College statistics group for assistance with the problem in appendix 1

I would also like to thank John Daly and Derek Coggle for technical support, Greg McCarra for keeping the computer running, Peter Beards for allowing me generous access to facilities at Napier College, and the Napier College library staff.

I would like to acknowledge financial assistance from the University of Peradeniya Sri Lanka, and the Peoples Bank of Sri Lanka during the first year of this work.

Finally, I would like to thank my parents and Véronique for all the help and encouragement they gave me over this period.

Contents

Prologue

This thesis is concerned with the problem of recognizing objects by computer. Humans and animals have had a highly developed sense of vision for many thousands of years, but the task of teaching a machine to 'see' has turned out to be more difficult than at first thought. Much research has been carried out to investigate the problem, and many techniques and systems have been designed.

What then is the motivation for research into computer vision? Why do we need artificial vision systems? Industry has a large demand for accurate and reliable sensing. Much of this sensing is done at present by humans. Visual inspection of finished products is a typical example. Such jobs are extremely tedious for humans, which results in a low performance level. Kruger and Thompson [1981] assess the need for computer vision in industry. They state (p.1524):

> "The economic motivation for the use of industrial computer vision is to increase productivity through the introduction of intelligent programmable vision-based systems for inspection and/or robotic assembly. Productivity is defined as the output of goods or services produced (or inspected) per unit of labour input."

They go on to quote Solow:

> "More than half of the increase in productivity [in the USA]

> is a residual that seems to be attributable to technical change, to scientific advance, to industrial improvements, and to improved management and training of labour."

Another area of application for computer vision systems is in environments that are unsafe for humans such as in power plants, or underwater. There is also the possibility of using vision systems that respond to a far wider range of electro-magnetic radiation than the human visual system. This may give new insight into difficult problems in many branches of physics and engineering. Vision systems could also be used to aid blind people with their everyday lives.

Therefore, there appears to be a vast demand for artificial vision systems that could perform as well as, or better than, the human visual system. Unfortunately though, the problem has been found to be of immense complexity, and this has motivated a large amount of research in the field.

This thesis looks at the problem of industrial computer vision. An architecture is developed for this environment, keeping in mind the possibility of application to other similar environments. Therefore, the vision system architecture is not designed to be limited to the industrial environment. The principal objective in designing this architecture was to attain flexibility of operation. The system is expected to be insensitive to a range of operating conditions with the aim of obtaining maximum flexibility coupled with a rapid execution speed.

Reader's guide to the thesis

Chapter 1 introduces the thesis. It begins with an overview of this work, which is followed by a brief survey of previous work in industrial computer vision.

Chapter 2 describes the general principles on which the new system is based.

Chapter 3 describes the architecture of my vision system. Detailed arguments are given for the design choices. This chapter is written in the form of a specification of the vision system, and therefore attempts to be independent of the actual algorithms used. The system architecture is then compared with that of previously reported systems with emphasis on performance.

Chapter 4 gives details of the implementation, the algorithms and data structures used, programming trade-offs etc. It is designed to provide sufficient information to allow the vision system to be implemented by the reader.

Chapter 5 reports the tests performed on the system to verify processing speed, and the degree of operational flexibility displayed.

Chapter 6 looks at the limitations of the architecture and proposes ways of removing them. In particular, this chapter suggests a way of extending the system to '3 dimensional vision' in the industrial environment. Next a strategy for implementing the pre-processor in hardware is given. This concludes the thesis.

Abbreviations and Conventions used in the Thesis

UF -Unique Feature

WTED -Walsh Transform based Edge Detector

The following conventions have been adopted in this thesis.

1. Quotations from other authors are always bracketed by double quotes as in " ... ".
2. Square brackets [ ] have been used to indicate references to other work.
3. Curly brackets {} are used to refer to material within this thesis - as in 'see section {1.1}'.

Chapter 1

Introduction and Survey of Techniques

This thesis is concerned with the problem of recognizing industrial objects rapidly and flexibly. The design objective was to attain operational flexibility in terms of minimum requirements placed on the operating environment coupled with a rapid execution speed using readily available processing resources. I will be especially interested in the problem of recognizing complex objects, i.e. objects which cannot be easily modelled by simple geometric shapes. An attempt has been made to keep the strategy fairly general, so that similar problems may be tackled using the same strategy requiring only a redesign of lower level algorithms.

The recognition strategy is based on the automatic learning of unique, reliable features of objects. Uniqueness of a feature is defined over the set of known objects, and reliability over the set of possible imaging conditions. Thus, a feature F is unique to object O_1 if (a) it is reliably located in the image whenever object O_1 is known to be in the image, throughout the complete range of imaging conditions that the system is required to operate in, and (b) if F is never seen in the image whenever object O_1 is known to be not in the image throughout the range of known objects and allowed imaging conditions.

Such a feature F may then be used (by definition) to reliably recognize an object from the set of known objects throughout the range of allowed imaging conditions. This is the strategy that is used in this thesis. (It is useful to note here that, in general, a feature may be a relational structure of other features, so that this algorithm can be shown to be a general requirement of any recognition strategy).

The vision system architecture is based on three main sub-blocks: (a) The use of generalized local features, (b) automatic learning, and (c) the use of unique structure. In the rest of this thesis I show the need for using generalized local features, and define what I mean by object structure, and how I select unique structure. The importance of defining the range of imaging conditions in advance is also explained. It is shown that the use of an automatic learning strategy results in a flexible recognition algorithm.

The system was tested over a range of imaging conditions. It was able to show insensitivity to its three main assumptions: constant lighting, constant scale, and limitation to 2D views of objects. For example, with one set of test objects, it was possible to reduce the light intensity by 70% before recognition was lost, or the object size could be reduced by 30%, or the object could be tilted 30°-40° out of the learned plane before recognition was lost. This performance was achieved despite the fact that the initial learning stage did not allow for variations in these parameters. A variety of other tests such as recognition of overlapping parts, recognition despite added Gaussian noise, image blurring, etc. demonstrate the flexibility of the system. In addition to this flexibility, the recognition algorithm was able to execute rapidly. Recognition times as low as 10ms

have been observed. However, average times when searching for 3 objects were from 100-500ms. For complex scenes execution times of 1-5s were reported. However, it should be noted that these times do not include a constant pre-processing time of about 70s which may be reduced to a negligible pipeline delay by the use of special purpose hardware. An architecture for such hardware is presented in chapter 6. It should also be noted that these execution times were obtained on a small minicomputer (PDP 11/24) programmed in Fortran.

1.1. Scope of this Work

In this thesis I will discuss only the problem of recognizing objects. I will not be concerned with the problems of symmetry analysis, or inspection, or measurement, or the problem of determining the position and orientation of the recognized object accurately. The reason for not discussing these issues is that I do not have any original contribution to make on these subjects. See Bolles [1979] and Olsztyn et al [1973] for a discussion of symmetry analysis. See the following references for a discussion of inspection: Brauner [1982] (IC chips), Baird [1982] (instrument gauges), Hara et al [1982] (printed circuit boards), Konishi et al [1982] (CCD wafers), Zimmerman et al [1982] (hybrid circuits), Barnard [1980] (industrial parts), Perkins [1983] (industrial parts). Also see PAMI [1983] which has a special section on industrial applications of machine vision; many of the systems reported are concerned with the problem of inspection.

I am also not concerned with the problem of scene analysis as it is normally understood, except for describing the image in terms of known objects. Therefore, the system does not attempt to explain the

light sources, shadows, or highlights etc. i.e. it does not attempt to account for all of the 'information' in the scene.

1.2. Industrial Computer vision systems

A large number of computer vision systems have been built, and reported in the last few years (Cohen and Feigenbaum [1982], and Raggett [1980] survey the field). A large proportion of these systems are concerned with the recognition and (or) inspection of industrial objects. In this section I look at the field of industrial computer vision in general.

The industrial environment is a popular choice for the design of computer vision systems (Chin and Harlow [1982] survey the field). Apart from the attractions due to economic factors (i.e. availability of resources), the industrial environment allows the vision problem to be highly constrained, and still be of use.

A large number of constraints are commonly imposed by computer vision systems, although not all of them are entirely acceptable to the average industrial user. The following is a discussion of these constraints.

1. The most important set of constraints is imposed by assuming a narrow context of operation. Objects will usually be presented to the system in a known way (e.g. on a conveyor belt). It is often assumed that only a single object will be visible to the system at any instant. Alternatively, some systems allow multiple objects provided that they are not touching. Others extend to touching objects, or to partially overlapping objects. The objects are usually seen on a uniform background. Some systems assume that the object is darker than the background or vice versa (but not both). The detection of object movement is usually not required, and many systems freeze object

movement (using hardware) before sensing the object. The recognition of object classes is also not required (e.g. the class of chairs, or tables). Objects are usually rigid and shape invariant (e.g. a half-open pair of scissors would not be allowed). Objects are usually not described in 3 dimensions, but as a set of views obtained from gravitationally stable states. This removes the need for 3D interpretation and representation.

2. It is common for computer vision systems to impose restrictions on the lighting conditions used. Some systems require special lighting conditions such as light stripes and light tables. Others use special lighting arrangements to highlight features known in advance.

3. Most vision systems assume that there is no scale variation i.e. the camera is fixed, and the objects are always at the same distance from the camera. Some systems show a tolerance towards small scale variations (e.g. Perkins [1978] ~ 5%). A further restriction is placed on the ratio of the largest to the smallest object. This is necessary due to the limited picture resolution available.

4. Most systems assume that the number of possible objects in the world of the vision system is small (of the order of 10).

5. Many assume that the objects presented contain a large proportion of straight and circular features, characteristic of man-made objects. Special feature detectors are often employed to respond to these features.

6. Some vision systems depend on assistance from a trained

operator during the object learning stage.

Of course, not all vision systems impose all of these constraints. The industrial vision environment in turn imposes special constraints on the vision system.

1. Low cost: Apart from the cost of building the vision system, the cost of providing the industrial environment must also be low (e.g., the cost of providing special lighting conditions, clean conveyor belts, etc.

2. Execution speed: The vision system must be able to perform at the required speed despite the constraints placed on cost.

3. Recognition must usually be achieved from a single view of the object(s).

4. The classification must be reliable. It may for instance be safer not to recognize an object (and therefore discard it), than to misclassify it. The requirement of reliability also forces vision systems to be less sensitive to the constraints they impose on the operating environment.

5. Some of the constraints that may be necessary for the vision system to operate could be unacceptable for an industrial user due to human factors. i.e., light flashes or lasers may be unsuitable if the vision system is to operate close to human workers.

6. Industrial users may be unwilling to supply trained operators to aid the vision system. Vision systems should therefore be

designed to operate with minimum human intervention.

7. It is likely that many of the objects to be recognized will have shiny metallic surfaces resulting in an increase of highlights in the image.

On the basis of the constraints placed on the operating environment, industrial vision systems can be divided into special purpose and general purpose vision systems. Special purpose systems are defined as those that seek to solve a specific industrial vision problem. These systems often use object dependent algorithms which are not easily generalizable to other tasks. Such systems are of limited interest to us. General purpose vision systems, on the other hand, are defined as systems that try to relax as many constraints as possible, and yet achieve the cost and speed requirements of industrial users. No system could hope to remove all of the constraints stated above (in the near future) as such a system would surpass the performance of even the human visual system. Therefore, the aim of a general purpose vision system is to remove as many constraints as possible, with priority given to those constraints that are expensive to satisfy. The vision system proposed in this thesis is such a system.

1.3. Survey of industrial vision systems

This section looks at general techniques used in industrial vision systems. It should be noted that the description of some systems that are directly comparable to the vision system described in this thesis is delayed till chapter 3 where they are discussed in greater detail. Thus, the purpose of this section is to give a brief overview of industrial vision techniques. The discussion begins with binary vision systems and progresses to grey scale vision systems. General techniques will be discussed along the way.

1.3.1. Statistical Vision Systems

Computer vision systems can be divided (loosely) into two categories: 'statistical' vision systems and 'structural' or syntactic vision systems. Statistical vision systems are essentially concerned with the classification of patterns using the well developed work in statistics and probability. The basic assumption is that the pattern generating mechanism (i.e. the scene and the imager) can be modelled as a statistical distribution [Devijver and Kittler 1982, p.6]. The recognition of a pattern then becomes a problem of statistical decision theory.

Thus the recognition problem can be defined as the problem of classifying an input pattern x to a single class C_r selected from a finite set $\{C_1, \ldots, C_n\}$, using a set of features $\{f_1, \ldots, f_m\}$. If the input pattern has a feature vector V, it is classified to be in class C_r if

$$D(V)-D(C_r) > D(V)-D(C_i) \quad \text{for all } i \neq r,$$

where D is the discriminant function. Many discriminant functions have been proposed. See Fu [1982], p.35 for a list. See also Devijver and Kittler [1982] for a detailed discussion of statistical pattern recognition.

This technique, although very successful in certain domains (see section {1.3.1.1}) of the vision problem, has two main drawbacks in terms of general vision.

Firstly, the discriminant function is essentially impartial towards its response to any given feature, and therefore minimizing $D(V)-D(C_i)$ does not guarantee a correct interpretation, especially in the presence of noise. This is because the D function essentially has no understanding of the physical importance of particular features in discriminating between objects. A common solution to this problem has been to use decision trees to reflect the importance of particular features. The decision trees are sometimes based on ad hoc programmer chosen criteria, which is unfortunate as the original reason for using statistical tests - that of a rigorous mathematical background - is lost.

Alternatively, a near optimal decision tree may be computed [Giralt, Ghallab, and Stuck 1979] by using Bayes decision theory to minimize the risk of misclassification. But, this requires knowledge of the multivariate probability function $p(x|C_i)$ when pattern x is known to belong to class C_i. This is sometimes computed empirically during an initial learning stage.

Another significant drawback with this technique is the inherent inability of statistical vision systems to analyze complex scenes, as

there is no statistical mechanism for handling 'structural' information. This is discussed in more detail after the next sub-section.

1.3.1.1. Statistical Binary Vision Systems Many binary vision systems that use statistical techniques have been reported (e.g. Agin [1975]). Objects were illuminated to produce high contrast images (e.g. by using a light table), so that the objects were easily separated from the background. A digitized TV image of this scene would then be segmented into object and background, and the statistical measures computed from the sensed image of the object. These measurements could then be used to recognize objects based on the heavy operating context. Measurements such as perimeter of object, number of holes in object, max/min moments of inertia were used. These systems operate well if the constraints placed are acceptable. Unfortunately though, this is not always the case. The requirement of back lighting can be problematic in the presence of conveyor belts. Reflected lighting can obtain the required lighting effects, but that places constraints on the background reflective coefficient, and on the stability of the incident light intensity. (Agin [1975] uses fluorescent red paint on the background, illuminated by ultra-violet light). Also, it is often necessary to use objects that do not fit into the assumed context, i.e. objects that are different (to humans), which generate similar measurements. This problem is often created by the two sides of a flat object. Further problems are encountered if touching or overlapping objects are to be recognized, or if dirt or swarf is present on the conveyor belt.

As with the simplest vision systems, the drawbacks of these binary vision systems arise from their inability to handle commonly occurring situations in industrial vision which do not fit the operating context.

1.3.2. Structural Systems: The need for an alternative approach

The success of a statistical approach is usually dependent on the selection of a good set of features. Although this may be relatively easy for simple scenes (such as when recognizing machine printed characters), it becomes quite difficult for complex scenes (such as when overlapping parts are present), or virtually impossible in very complex situations (such as in a street scene). The reason for this is that as the number of possible objects and the range of imaging conditions is increased, the number of pattern classes explodes rapidly, and it is no longer possible to treat the problem as one of pattern classification.

However, the problem may be tackled using a structured approach, by treating the scene as consisting of several subparts that are related to each other in some way. It is now possible to treat each subproblem as a pattern classification problem. Indeed, Devijver and Kittler [1982] p.3 assert that for the majority of problems "either the original problem itself can be reformulated as a pattern classification problem, or it may be divided into a number of classification subproblems and sub-subproblems until, eventually, the original problem is reduced to a set of pattern classification problems".

For example, the recognition of a circuit diagram is a problem that cannot be treated as a straightforward classification problem. It could be handled by treating the recognition of individual components (such as the resistors and capacitors) as a pattern classification problem. Once this is done, the circuit diagram can be represented as a relational structure of subparts that have been recognized. The analysis of the circuit can be continued from this point.

In addition to the objective of reducing the recognition problem to a set of (not necessarily independent) subproblems, structural systems also have the objective of describing the physical structure of the objects. However, current usage of the term does not appear to insist on this. One of the reasons for this is that it is very difficult to define what is meant by physical structure of the objects.

1.3.2.1. Shape descriptors A commonly used attribute of an object that is accepted as reflecting its structure is the shape of its boundary. A popular shape descriptor is the chain code, first suggested by Freeman [1961]. In the general version of this technique, the local direction of the boundary is quantized to one of a finite number of directions. Each segment is then linked to its nearest neighbour, to form a chain of edge segments. Kopolowitz [1981] investigates the performance of chain codes. Mckee and Aggarwal [1977] use an extended chain code to recognize partial views of objects from binary images. A feature of the system is its ability to handle scale variations.

Karg and Lanz [1979] represent shape using concentric circles centred on the centre of gravity of the object. Olympief et al [?] represent shape using a polygonal approximation. Pavlidis [1978,1980]

surveys the multitude of shape descriptors that have been reported.

1.3.2.2. Structure from Local Features Another approach to improving vision system performance is to use local features that reflect the structure of the object. Igarachi et al [1979] reports a special purpose vision system for integrated circuit (IC) wire bonding that has a special IC electrode detector. The system improves its noise immunity by using a dynamic threshold to obtain the binary image. The advantage of local feature detection is the ability to withstand a certain amount of obscuration or noise which affects global features such as area. Persoon [1978/9] uses local information to allow his binary vision system to recognize touching, or partially overlapping objects. In the learning stage the system learns local binary shape patterns of 11 pixel diameter. The binary vision system reported by Bolles and Cain [1983] is able to use local features such as holes and corners.

1.3.3. Other binary vision systems

Kelley et al [1979] describe a vision system that uses binary vision to pick objects from bins. This system is interesting in that it is a rare example of a computer vision system that uses tactile information and its grasping ability to aid the recognition process. The vision system is used initially to decide on a suitable site for the robot to grip. The robot then tries to pick up the object using its tactile sense to detect error conditions (or success). Once the object is picked up, it is shown to the camera on a suitable background so that it can be recognized and oriented.

Taylor and Ero [1980] report an unusual vision system that performs a complete 2D correlation of the input image with all of the stored images simultaneously, using special purpose hardware. The system is able to operate at a speed of 125 objects/s. A drawback with the system is the need to store different 2D orientations of the same object as different models. It should be noted that direct correlation techniques (also called template matching) have another drawback in that they require the object position to be the same in the image as when it was first taught. In Taylor and Ero's system, this problem is circumvented by the use of a conveyor belt which effectively sweeps the object over one of the axis. The object position has to be accurate on the axis perpendicular to the direction of motion of the belt.

1.3.4. Structured lighting for binary vision

The CONSIGHT vision system of Ward et al [1979] is a statistical binary vision system that uses a few global measurements of the object for recognition. However, they circumvent one of the problems of binary vision -the thresholding of the image to separate the object from the background- by using a unique lighting system based on using two planes of light which are focussed on to a thin strip of the background. This strip of light is then observed by a line scan camera. As the light planes are projected from non-perpendicular angles, any object with a significant thickness displaces the light stripe from the view of the camera. Object and background are separated reliably by this method, although some problems are introduced by a shadowing effect which can be minimized by careful setting up of the lights.

1.3.5. Binary/grey scale vision systems

A major disadvantage with binary vision systems is their inability to operate as the contrast of the input image is degraded. This has created the necessity to quantize the intensity to more than two levels, in order to increase the information content in the image. Some vision systems rely on a hybrid system that uses binary and grey scale images to improve the system performance. Malinen and Niemi [1979] report a system that uses a binary image and an 8 level (3 bit) grey scale image for object recognition. They assume the objects are dark when compared with the background.

Yachida and Tsuji [1977] report a sophisticated general purpose vision system. Objects are first located in the image using a coarse binary image. Once located, a fine grey scale image of the object area is obtained. This image is thresholded using a local histogram technique, so that local intensity variations can be accounted for. The resulting silhouette is classified using statistical measures and a shape descriptor based on the distance of points on the perimeter from the centre of gravity. The most likely matches found are then tested for, using special feature detectors to find holes, lines, small holes, and textures. The special feature detectors operate rapidly as they are used only over the local area where the feature is expected. The system contains a special learning algorithm. During the learning stage, all matches of the new object with library objects are tested for on the basis of the statistical measures and the silhouette shape. The special features necessary to distinguish the new object from the subset of matched models are taught by an operator during an interactive learning session. The system is reported to be able to

operate "with considerable noise caused by dirt and grease" and reports a fair operating speed (20-90s on a PDP8 minicomputer. This can probably be reduced by an order of magnitude or more on a modern (1984) minicomputer).

This versatile vision system suffers a few drawbacks as far as general purpose vision is concerned. It assumes that objects are brighter than the background, that the background area is larger than the total object area, and that operator help is available to teach local features. It is not designed to be able to handle touching or overlapping objects, although they report that "even when there were overlapping objects in the scene, the vision system could tell their locations ..." which implies a degree of operational flexibility. The system also suffers from an abundance of heuristics and programmer chosen weighting criteria. However, it appears to be superior when compared with standard binary vision systems.

1.3.6. Grey scale vision systems

Many of the problems associated with binary vision systems are due to the initial loss of information when the image is thresholded. Thus, it is necessary to use the grey scale image itself for the image analysis. One of the best known general purpose vision systems that uses grey scale images for object recognition was reported by Perkins [1978].

The program first finds edge points in the input grey scale image. This is necessary in order to reduce the data in the input grey scale image. Perkins uses a 256x256 input image. The edge detec-

tion reduces the initial data of over 65000 intensity points to "less than 1000 edge points". This is possible due to the large redundancy in most scenes. The edge data is then thinned and linked into chains. The thinning operation is necessary because most edge detectors produce edges more than 1 pixel thick (especially in the vicinity of a strong edge). The chains are formed by connecting edge points to their neighbours. The program now uses the a priori knowledge that most industrial objects have straight and circular features, to find these features in the chain data. The chains are therefore transformed into a set of linked segments that are either straight or circular. These new chains are called concurves. Models of the objects are formed during a learning stage by storing the concurves found. In the recognition stage, model concurves are compared with input concurves. The program is claimed to be able to operate in visually noisy scenes and is able to recognize overlapping objects (although it was not designed to do so). The program reports a rapid execution speed (on an IBM 370/168 mainframe computer) of between 0.1s and 0.4s for the high level operations; the low level algorithms taking approximately 20s. It is limited to the recognition of stable states of objects (i.e. no 3D interpretation is attempted), and it cannot handle textured object data. It is also dependent on finding straight and circular features in objects in order to operate efficiently. The program is not able to handle scale variations either, although a 5% variation is tolerated. (Shirai [1978] reports a similar system that uses straight lines and ellipses to recognize "everyday objects" in 3D scenes).

1.3.7. Planes of activity

Perkins' program illustrates a special instance of the strategy used by computer vision systems. The input data is reduced and transformed into a set of features that are independent of certain imaging conditions such as absolute lighting level, object position and orientation, and scale variations.

Pre-stored models of the expected objects are then used to select (or generate) a set of possible features. The features found in the image are then compared with the features generated from the models. If the two sets are sufficiently similar, the object is declared recognized. We then identify three planes of activity. The image plane, the feature plane, and the model plane (see Fig. 1-1). These planes of activity have only a loose association with the levels of processing that are commonly referred to in vision research (i.e. low, high and intermediate level vision).

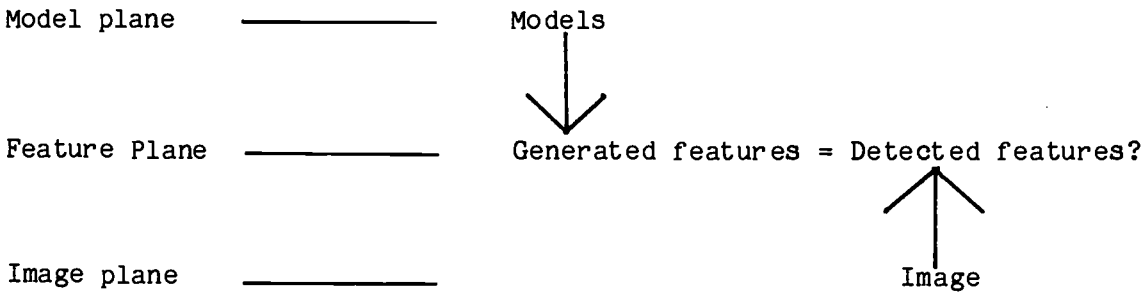


Fig. 1-1

In Perkins' program, there is hardly any distinction between the feature plane and the model plane. During the learning stage, features that are found are used directly to form models. Because the features formed are dependent on the 3-dimensional orientation of the object, each 2D view of the object generates different features (in general). Thus, Perkins' program is limited to the recognition of stable states (i.e. a set of 2D views) of objects (as are most industrial vision systems). Yachida and Tsuji's program {section 1.3.5} can be classified into the three planes in a similar way. However, their program has a feature plane that is quite wide in terms of the level of processing used in the feature match.

The ACRONYM vision system by Brooks et al [1979] is a good example of a vision system that has 3 distinct planes. The model plane contains 3-dimensional geometric models of the objects that are known to the system. These models are given to the system by an operator. The system then predicts the features that it expects to see in the scene. These features are matched with features found by the low level algorithms. The features used are "ribbons" (2D projections of generalized cones) and ellipses.

The system by Taylor and Ero {section 1.3.3} is an interesting example. Models are formed by storing the input binary image. In this system, the three planes coincide.

1.3.8. Summary

Computer vision systems, then, differ by the way they use features and models. Some systems use pixel data as models and there-

fore can recognize an object only if it is presented in exactly the same way as it was when the model was formed. Flexibility of operation is achieved by extracting features of objects that are invariant with respect to the imaging conditions of interest. These features can then be compared with previously stored features. However, it is not possible to extract a set of invariant features with respect to certain parameters such as 3D orientation. In order to handle such situations, it is necessary to form 3-dimensional models of the objects, so that the expected features can be predicted and searched for.

Chapter 2

A Strategy for Recognizing Complex Objects: Basic Principles

In this chapter I describe the strategy that was developed to tackle the problem of recognizing complex objects in a flexible manner.

2.1. The General Strategy

The recognition strategy is based on three building blocks:

(1) generalized local features,

(2) learning, and

(3) the use of unique structure.

The system operates as follows: During the learning stage, the generalized feature detector generates feature descriptions of the objects to be learned. The learning algorithm observes the performance of the feature detector and selects a set of reliable features for each object. From this set, the learning algorithm constructs a set of descriptions of unique, reliable subparts of each object. During the recognition stage these unique descriptions are searched for in the new feature description.

I will now describe the three sub-strategies in more detail, to dis-

cover the extent of their generality.

2.1.1. Generalized Local Features

The task of the feature detector is to describe the input scene using a set of features so that all of the information is included. The system describes the scene at two levels. Firstly, the scene is described by a relational structure of elementary features called rep-points {section 3.2.3}. Next, local subgraphs of rep-points are used to form local features. The scene is then described by a relational structure of local features. Thus the local features are local subgraphs of the rep-point relational structure. These local features are able to describe complex local structure of the objects due to the way they are constructed. The large vocabulary of the feature descriptor {appendix 1} makes it a generalized local feature descriptor.

Thus, a major objective in the design of the feature descriptor was to allow it to describe object structure in detail. What then is object structure, and why is it necessary to describe it in detail?

2.1.1.1. The need for a better description of object structure

Object structure is very important for recognition. Since the structure of an object is constant through variations in imaging conditions, a vision system that is able to respond to object structure would be very successful. But what is object structure, and how can it be defined?

Object structure is relative. Firstly, it depends on the scale of interest; my vision system, for instance, will not be interested in the internal structure of objects. This is effectively a requirement that the structure of the object be visible. Secondly, what is generally referred to as structure depends on the context of use; the perceived structure of an object is often dependent on the perceiver and his motivation. (Consider, for example, the perception of circuit diagrams, chest X-rays, weather photographs, hand writing in a foreign language, etc. by people trained to do so, and the rest of us). Thus it appears that we need a definition of structure that is independent of human perception, but is useful for computer vision.

The previous two paragraphs in fact provide us with the required information for a definition of (visual) structure. From the first paragraph it is clear that the motivation for using object structure arises from its independence of imaging conditions. From the second paragraph, the important condition is that the structure must be visible. Therefore, I define object structure as everything about the object that is independent of the imaging conditions of interest, and is visible. Imaging conditions are defined as everything that contributes to the function that transforms an object into an image. In addition to parameters such as lighting and scale, it also includes lens aberrations and electronic noise.

The reason for the qualification on imaging conditions (to that of the imaging conditions of interest) is due to the fact that the set of visible features that are independent through (all variations of) the imaging conditions is of course a null set (e.g. there must be limits placed on the allowed variation of lighting, scale, etc.). The

reason for requiring 'everything' about the object to be structure is because my system will depend on structure to differentiate objects, and therefore, any two objects that have the same structure (as defined) will be indistinguishable. It is therefore necessary to respond to everything that might differentiate the two objects. Thus, this requirement means that two objects can be reliably differentiated only if they never produce exactly the same image within the imaging conditions of interest. It will be noticed that this is not restrictive, and is in fact a fundamental principle of vision when external contextual information is unavailable.

From this definition it will be clear that all of the information regarding the object structure must be present in the image. The objective of the local feature detector is to describe the local structure in a form that is independent of the imaging conditions of interest. Therefore the feature detector attempts to respond to everything in the image that is independent of the imaging conditions. In particular, the local feature detector is not designed to be limited to those features that are thought to be important by the human visual system.

2.1.1.2. The advantage of using local features The feature detector is designed to detect only local features, for two reasons.

1. Local features are less sensitive to object obscuration.
2. Global features can be constructed from the local features.

The first of these is the main motivation for using local features. It allows us to recognize partially visible objects using local feature

descriptions of the visible part.

2.1.2. Learning

The heart of the general strategy lies in the learning system. The learning strategy is responsible for the performance of the system. It improves the speed, the flexibility, and the reliability. The learning stage has the following tasks:

1. Acquire a description of each object in terms of a relational structure of rep-points, and a relational structure of local features.
2. Observe the performance of the feature detector and form a set of reliable features over the imaging conditions of interest. Obtain insensitivity to internal parameters of the system as well by this reliability test.
3. Compare the objects that have been learned, and find feature descriptions of subparts of each object that are unique to the object over the imaging conditions of interest., and thereby transfer the graph matching problem from the recognition stage to the learning stage. These descriptions of unique subparts of the object are called unique structure.

2.1.2.1. Unique Structure

Let us imagine that we have a perfect pre-processor that is able to describe objects in terms of features that include all of the information in the scene that is independent of the imaging conditions of interest, as required in section {2.1.1}. These features will be called structural features (see definition of object structure {sec-

tion 2.1.1.1}) to differentiate them from features which we do not know are independent of the imaging conditions. Each object will then be represented by a set of structural features, and since all of the structural information is included in the feature description, the objects can be differentiated on the basis of the feature description alone i.e., any two objects that are indistinguishable from the structural feature description are visually indistinguishable over the imaging conditions of interest. (Note that this is for a perfect pre-processor). How can we compare the feature descriptions to recognize objects?

It will be clear that the features have relationships between them. That is, it is insufficient to detect each feature in isolation. The complete feature description is required to specify the complete object. Therefore, the object must be represented by a relational structure of features. (It should be noted that at this stage of the discussion the word feature is used to describe any feature that one would want to measure, and is not limited to those used in this work, or to local features). The task of object recognition then becomes a problem of comparing (or matching) relational structures. The basic problem here is the one of matching two graphs. Much work has been done on the graph matching problem, and so we digress here to look at the problem and how it has been tackled by other researchers.

2.1.2.2. Relational structures and graph matching

The structural method for representing object model data is as a relational structure of sub parts of the object

[Ambler et al 1975]. The comparison of objects for recognition then reduces to a problem of matching relational structures, i.e. of graph isomorphism. Unfortunately, no general and efficient algorithm is known for testing isomorphism of large graphs (i.e. graphs with more than about 10 nodes. Unger [1964]). This has resulted in a number of special techniques for reducing the execution time. Unger [1964] gives a heuristic algorithm. Ullman [1976] reports an algorithm that takes time proportional to p^3 where p is the number of nodes in the graph. The algorithm by Corneil and Gotlieb [1970] takes time proportional to p^2. However, these algorithms work best on certain classes of graphs. Corneil and Gotlieb's algorithm, for example, is inefficient for strongly regular graphs.

Ambler et al [1975] match relational structures by setting up a new graph G whose vertices are formed from matching nodes in the two relational structures. The edges in G link "compatible" vertices. Vertices in G are compatible if the transformation implied by matching nodes of the relational structure are the same. The problem of relational structure isomorphism then reduces to that of finding maximally connected subgraphs (cliques) in G. They give an algorithm to find cliques similar to that of Bron and Kerbosch [1971]. Osteen and Tou [1973] report a recursive algorithm for clique detection based on neighbourhoods in graphs.

Cheng and Huang [1981] reduce the problem of relational structure isomorphism by using "star-structures". A star-

structure is a sub-relational structure of a node and all of its neighbours. Relational structures are matched by setting up a graph G of matching star-structures (as above) and finding cliques. They find cliques using a relaxation algorithm that converges rapidly (10-20 iterations). Cheng and Huang [1982] are able to use this technique to extract motion information by image registration. In the example given, the algorithm executed in 24.5s on a PDP11/70 for a 70 node graph.

Jacobus and Chien [1979] describe a system that matches graphs of "half-chunks" to determine recognition. A half-chunk consists of two line segments and a tangent angle. They convert object graphs to histograms by recording the number of occurrences of library half-chunks in the object. Objects are matched by comparing histograms. The reliability of the histogram matching technique is not clear.

Thus, the problem with graph isomorphism is that it is computationally very expensive. In my vision system, the problem would be to find subgraph isomorphism of graphs with as many as 400 nodes. (Even more for complex scenes). Graph isomorphism has another, very important problem. Let us imagine that we have two object models, each of which is a relational structure of 400 nodes. Now let us also imagine that we have a new image from which we have formed a new graph of 500 nodes. After exhaustive graph isomorphism, let us imagine that we recognize a subgraph of 250 nodes of the first object in the image, and a subgraph of 300 nodes for the second object, with 100 of the nodes being common. Which object did we recognize? Are both objects in the image? How different do the numbers have to be before we choose one object

over another? What principles do we have in choosing thresholds? What effects did noise, and a variation of imaging conditions have on these figures? These questions are difficult to answer, but they illustrate that graph isomorphism is only half the problem. It is necessary to interpret the result from the graph match.

This brings us to an important point. Similarities between objects only serve to confuse the final decision. The fact that we have recognized 50 features that are common to both objects tell us nothing about which object is in the image. This is of course a fundamental principle of recognition; It is not possible to recognize object A from object B from their similarities!

It is therefore very important to know what makes one object different from another. Once again, let us indulge in a thought experiment. Imagine that we have two objects A and B with 200 features each. Imagine that 150 of these features are common to the objects (but not exactly the same) i.e., these features are close enough to be confused by the feature detector. Now, it will be clear that the remaining 50 features of each object are essential for recognition. The problem is this: Most vision systems match objects at recognition time without prior knowledge of the similarities and differences between objects. Therefore, they use the 150 similar features as well as the other features to base their decision. Now let us see what happens when we have a noisy input image. Because the similar features are not <u>exactly</u> the same, the 150 features from the image (C) may match object A features better than object B features. (See Fig. 2-1)

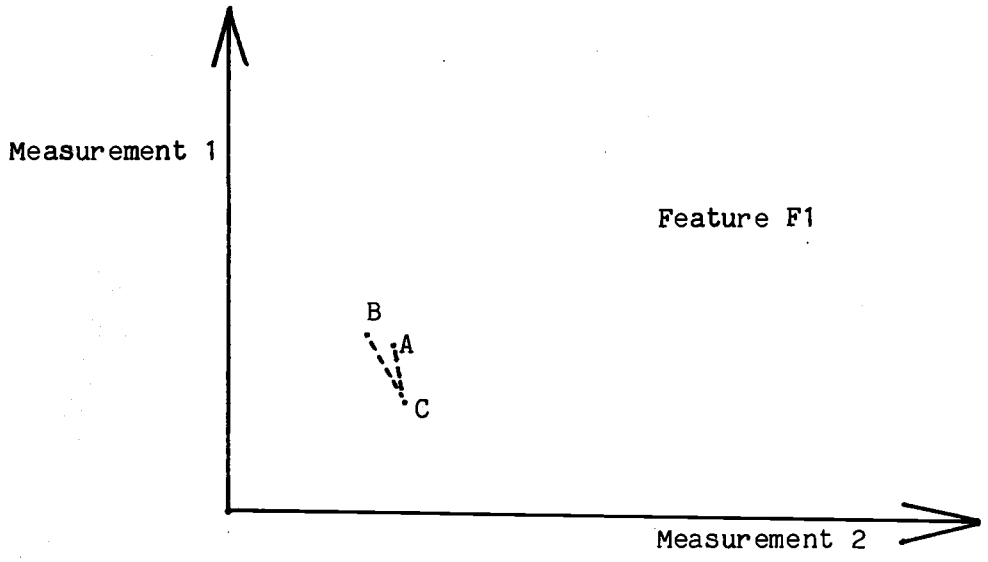


Fig. 2-1 A single 'similar' feature represented in 2D parameter space

Now it is clear that fluctuations in the number of features matched for the similar features can swamp the number of dissimilar features matched. In an extreme case, it is conceivable for all 150 of the similar features in C to match those in A, but not match those of object B, while all the dissimilar features continue to match those in B. This would result in a mismatch. However, a system which knew that the 150 similar features in A were in fact very close to the other 150 features in B, would not use this information to differentiate between A and B, and would place far more emphasis on the 50 dissimilar features being matched.

Thus, the requirement placed on my system is to find similar and dissimilar structure. (Since structure must be constant through imaging conditions, dissimilarity must also hold through variations in the imaging conditions. It is insufficient for two features to be dissimi-

lar under a single set of imaging conditions.) When the number of objects is increased to more than two, the requirement is to find unique structure for each object i.e., find out what makes an object different from the rest of the objects over the imaging conditions of interest.

Thus, my vision system recognizes objects on the basis of their unique structure. Is this restrictive? Let us look at the definition again. The unique structure of an object is everything about the object that is

1. visible,
2. independent of imaging conditions, and
3. different from the other objects.

Therefore, if an object has no unique structure, it must be visually indistinguishable from at least one other object in the learned set.

2.1.2.3. Finding Unique Structure The first step is to describe the objects in terms of a set of elementary features (that contain all the information). However, these features do not become structural features until it is verified that they are independent of the imaging conditions of interest. Once a set of structural features is selected, the objects are represented by a relational graph of these features. Unique structure is found as follows: First form a set of subgraphs for each object by choosing all combinations of all features (and their relationships) so that each subgraph may have from 1 to N features, where N is the total number of structural features in the object. Now compare these subgraphs with those found in the other objects through the imaging conditions of interest (i.e., the

structural feature subgraphs are compared with subgraphs formed from all features found in each separate instance of all of the other objects.) Those subgraphs that do not match any in all instances of the other objects, form the set of unique structure. Some comments are in order.

(1) It will be noticed that this is a huge task as there are rather a large number of subgraphs (2^n for n nodes). This may be limited artificially using arbitrary rules such as limiting the search to the 50 smallest unique subgraphs. (In the implementation, subgraphs are limited to local subgraphs. The locality is defined by the spatial distance between nodes. These local subgraphs are the local features detected by the generalized local feature detector.) Note that the time taken to do this is not critical, as it is done during the learning stage.

(2) A very important point now is that the recognition system has to search only for these unique subgraphs. Firstly, this makes the searching much faster. Secondly, since we know that the unique subgraphs are unique throughout the imaging conditions of interest, there is no danger of a mismatch. Thirdly and most importantly, the detection of a single unique subgraph is (by definition) sufficient for recognition. In practice, however, because it is not possible to guarantee a perfect reliability test at the learning stage, more than one subgraph is required for confirmation.

This strategy of finding unique structure illustrates another reason for requiring the local features to be highly descriptive. It

increases the likelihood of each feature being unique to the object.

Thus, the learning strategy results in

(1) improved recognition reliability due to the feature reliability test,

(2) improved speed due to the transfer of the graph matching problem to the learning stage, which reduces the recognition search to a single unique subgraph, and

(3) improved flexibility due to the extra unique descriptions produced by the learning algorithm, so that the system is able to operate despite the loss of a large number of unique subgraphs due to object obscuration or degraded operating conditions.

In addition to these advantages, there is a fundamental need for learning when flexibility is required. Let us imagine that we require a vision system to be able to recognize objects despite a 10% variation in object size. This of course means that it is not possible to differentiate between two objects that are only 10% different in size (even if we wanted to). This is unacceptable for a 'general' system. It is more likely that we would require that under such a situation, the vision system should automatically reduce its flexibility to (say) 5% for the two objects of concern, and retain a flexibility of 10% for the other objects. Further, in order to obtain maximum flexibility it would be useful for the system to adjust its flexibility upwards when the objects are very different.

Such variation in flexibility is clearly present in human performance. We are able to distinguish (say) a house from a man despite a significant amount of image degradation, but we can tolerate less

image degradation if we are to recognize one man from another, or one 'identical' twin from another. Such a variation of flexibility may be achieved by allowing the system to learn about the similarity between the objects of interest.

Another objective of the overall design was that the system should be insensitive to the actual details of the implementation of the architecture. That is, the system was expected to function properly despite minor imperfections in implementation. This insensitivity is obtained by using the extended learning strategy which compensates for pre-processor imperfections by observing its performance, and rejecting features that are not reproducible, either because the feature is dependent on imaging conditions, or because the feature is distorted by the feature detector by being associated with a non-linear section of the feature descriptor mapping function.

The extended learning capability of the vision system is therefore responsible for (1) improving the reliability, (2) improving the speed, and (3) improving the flexibility of the system. It is felt that a learning strategy that is able to do this is of general interest.

2.1.3. The Overall Strategy

The strategy then is to describe objects using features that represent all of the information in the image that is independent of the imaging conditions of interest. Therefore, the features are expected to have an extensive vocabulary, so that complex object structure can be represented. The learning stage consists of finding

unique structure for each object. Unique structure is a set of relational subgraphs of structural features that are unique to the object in question. A structural feature is a feature that is invariant through the imaging conditions of interest.

Thus, the system depends mainly on these principles. There is little emphasis on problems of detail such as threshold selection, feature matching etc. That is, the system is expected to operate well despite a possibly non-ideal selection of feature types, or thresholds, or matching criteria. The primary objective is to attain operational flexibility and speed of operation using these three principles, and a fairly good feature detector and feature matching algorithm. Thus the performance of the system is attributed to the exploitation of these ideas than to carefully worked out details of the system implementation. For this reason it is felt that the overall performance of the system could be improved by re-working the detailed design using information theory, empirical test data, and other considerations.

The strategy can also be expressed as follows:

Given a set of objects

$$A_1, A_2, A_3 \ldots A_n$$

that are to be learned, image each object i times <u>where i is large</u>, and obtain images

$$A_{k1}, A_{k2}, A_{k3} \ldots A_{ki}$$

for each object A_k over a set of imaging conditions IC.

Form descriptive feature sets FA_{kj} for each image A_{kj} by using a

feature descriptor. (Note: a feature may be a relational structure of other features).

Now for each object A_k form two sets of features $A_{k\cup}$ and $A_{k\cap}$ such that

$$A_{K\cup} = \bigcup_{\forall j} FA_{kj}$$

and

$$A_{K\cap} = \bigcap_{\forall j} FA_{kj}$$

Then, $A_{k\cup}$ is the set of all possible features for object A_k, and $A_{k\cap}$ is the set of structural features.

Now a set of unique features A_{k*} is formed as follows:

$$A_{k*} = A_{k\cap} - \bigcup_{\forall m\neq k} A_m$$

(i.e. for 3 objects B,C,D, $B_* = B_\cap - C_\cup - D_\cup$)

The recognition strategy is based on the following two properties of A_{k*}: If an image I is taken, and we form a new set of features FI, then,

$$A_{k*} \subseteq FI \qquad \text{---------- (1)}$$

when A_k is visible in the image, and

$$A_{k*} \cap FI = \phi \qquad \text{---------- (2)}$$

when A_k is not visible in the image.

This is <u>always</u> <u>true</u> for large i over the set of imaging conditions IC. This is the principle that is used in this thesis. The learning is concerned with the generation of sets A_{k*}, and the preprocessor is concerned with the generation of the features sets FA_{kj} during learning and FI during the recognition stage. Recognition is concerned with the verification of equation 1 above.

Chapter 3

A Strategy for Recognizing Complex Objects: The Architecture

This chapter describes the detailed architecture that was developed to exploit the principles set out in the previous chapter. I have tried to keep this material as independent as possible from the implementation details so that the strategy of the architecture becomes clear. For this reason, this chapter is written in the form of a specification of the required system with little or no mention of the actual algorithms and data structures used.

The chapter is organized as follows: After the introduction, section 3.2 describes the architecture of the pre-processor. Section 3.3 looks at the feature matching algorithm. Section 3.4 describes the learning stage, and section 3.5 the recognition stage. Section 3.6 looks at the overall architecture and makes a few general points, and finally, section 3.7 compares this architecture with previously reported architectures.

3.1. Introduction

The main objective {section 1.0} was to design an industrial object recognition system that is able to operate flexibly and fast

using the principles set out in the previous chapter. However, there were other less important objectives.

3.1.1. Other Objectives

1. In addition to being insensitive to operating conditions, the system was expected to be insensitive to internal operating details. For instance, a major requirement was that the thresholds used be static (unless it was possible to provide dynamic threshold variations within the available hardware resources). Therefore, all of the thresholds described in this thesis are for system development purposes only, and remain fixed when the system is used. Initial threshold selection was dictated by hardware resource limitations, empirical tests, and on theoretical considerations when possible.
2. The system was expected to be insensitive to minor imperfections in implementation.
3. It was felt that the problem of recognizing complex objects had not been properly addressed before. The objective was to use the complexity of these objects to advantage by employing a generalized feature detector to respond to these features. In fact, it was felt that no restriction should be placed on the complexity of the object geometry (or on surface markings etc.), apart from the unavoidable restriction due to image resolution.
4. The system architecture was to be designed so that it could be implemented on dedicated hardware with minimum effort. For this reason, the simplicity and parallelism of (especially) the low level algorithms was an important requirement.

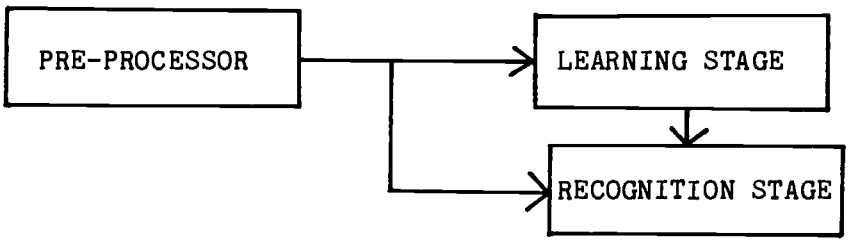
3.1.1.1. The Importance of Execution Speed It should be noted that execution time is of great importance in vision work even though one may feel that it does not really matter as long as the system works, as it is always possible to increase the execution speed by using better hardware and optimized code. However, vision algorithms can easily take exponential time to execute. Clearly, such algorithms become unmanageable very quickly, and may become unusable without some optimization. Algorithms that take long to execute become difficult to test thoroughly, using different image data, and different parameter values. It is my experience that overall execution times of more than a few minutes make systems very difficult to test properly. Therefore throughout this thesis execution times will be considered as an important parameter with which to judge vision systems, with the understanding, of course, that execution times are dependent on the hardware that is used.

3.1.2. Influence of the human visual system

It should be noted that during the course of this work, inspiration was drawn from published work into the human visual system. The interested reader is referred to Athukorala [1985] for a discussion and list of references to some of this work.

3.1.3. The System

The vision system consists of three main parts. The preprocessor, the learning stage, and the recognition stage.



The rest of this chapter will discuss these components in detail.

3.2. Pre-processing

The task of the pre-processor is to describe the input scene in terms of local features that are insensitive to imaging conditions. The pre-processor performance is not expected to be perfect. The learning algorithm is responsible for monitoring the performance of the pre-processor.

Ideally, the pre-processor would generate features that contain all of the information about the object that is independent of the imaging conditions. In other words, we would like the features to be

(a) invariant through operating conditions, but

(b) very sensitive to variations in object structure.

Further, the ideal pre-processor would be expected to display 'monotonicity' in its mapping function. What does this mean?

Let us assume for the moment that the objects we need to recognize can be completely specified by a single 1 dimensional feature. For example, our problem may be to recognize a set of thin iron rods which vary only by their length. The chosen feature would then be the length of the rods. We would then require that the pre-processor be able to describe this feature so that

(a) the length measured for a given rod is constant despite variations of the position of the rod, and variations of other imaging conditions, and

(b) that the length measured should vary, when the actual length of the rod is changed.

The requirement of monotonicity means that the measured length should increase when the actual length increases. Clearly, linearity would

be ideal, but that would be far too ambitious and demanding of a general purpose pre-processor. Monotonicity is a less demanding requirement. However, the requirement that is placed on my pre-processor can be relaxed further to that of being single valued in parts, due to the learning strategy. Therefore, the pre-processor is expected to deliver a subset of features that are independent of the imaging conditions, and are on part of the mapping function that is locally single valued. The learning algorithm will seek out the rest and reject them. To restate: the pre-processor is expected to describe similar structure using similar descriptions, and dissimilar structure using dissimilar descriptions. The learning algorithm verifies this behaviour over the subset of structure that has been learned.

3.2.1. Imaging Conditions

What are the imaging conditions that can vary in the restricted industrial domain I have chosen, and what are the imaging conditions of interest? Firstly, the definition of imaging conditions: I define this as everything that contributes to the function that maps object to 2D image. This includes camera view-point, lighting, lens parameters, camera electronics, light defraction, lens aberrations, etc. Clearly, we cannot hope to take account of the complete range of values that all of these parameters can take. The following is a detailed discussion of those imaging conditions that the system expects will vary, and the limits of these variations the system expects to cope with. The rest of the imaging conditions are assumed to be constant, and if not, the system would only expect to be able to tolerate minor fluctuations. (The system was in fact tested with

variations of two of these parameters: camera focus and added Gaussian noise. See chapter 5.) In the following, I will refer to a plane (P) which is defined to be perpendicular to the camera viewing axis (Z) and at a distance equal to the expected distance from camera to object (i.e. the plane of the table top or conveyor belt. Object height is assumed to be small compared with the distance from the camera to the object.)

3.2.1.1. Object position (2D and 3D) Clearly, it would be desirable for the vision system to be independent of the 2D position of an object within the visual frame i.e. the feature description generated should be invariant with object position provided the object is in view. Further, the feature description generated should be invariant with 2D object position provided the feature is in view (i.e. even if the rest of the object is hidden). Object position variations in 3D (i.e. when the distance to the camera is changed) changes the scale of the object and blurrs the image. Image blurring is a complex function of imaging conditions, and depends on the camera aperture etc. The system is expected to be resistant to a small degree of image blurring. Resistance to scale changes is discussed below. (Note: 2D position is the position of the object on plane P, and 3D position is its position on the Z axis - see earlier definition of P and Z.)

3.2.1.2. Object orientation in 2D (i.e. rotation about an axis parallel to Z). The system is expected to operate independent of the 2D orientation of objects, similar to the requirements for 2D position. The feature description must be independent through all values

of the orientation of the feature, (and therefore the 2D orientation of the object). These two requirements are achieved by describing the features using coordinate axis defined on the features themselves.

3.2.1.3. Illumination variation The system should be insensitive to slow variations in absolute illumination level. Illumination variations can occur due to many reasons.

(1) Variations in total intensity.

(2) Variations in lighting direction

(3) Variations due to shadows, highlights, or reflected light (i.e. mutual illumination).

Clearly though, it is not possible for any system to operate throughout the full range of values that some of these parameters can take. I therefore introduce the notion of 'reasonable lighting conditions'. Reasonable lighting is defined as the lighting conditions that would normally be provided for a human to perform the same task. For example, a human performing an assembly task would be provided with constant, bright lighting. S/he would not be expected to work in an environment with, say, flashing lights, moving light sources, semi-darkness, or blinding brightness. In fact, economics would probably dictate that it is cheaper to provide 'constant' lighting (with say, ordinary fluorescent lamps) than to provide flashing or moving lights. This then is thought to be a reasonable condition to impose on the industrial environment. If the system was required to operate in extreme conditions, it would be cheaper to provide special lighting conditions (for example by using a narrow bandwidth source and a narrow bandwidth filter on the camera to blanket out variations), than to

compensate for such variations using sophisticated general purpose software routines.

However, care must be taken when designing with 'constant' lighting in mind, as lighting thought to be constant by humans may not be constant 'enough'. (This is partly because we are relatively insensitive to absolute quantities, and because the visual receptors have a logarithmic response [Cornsweet 1970, p. 249]). Further, it would be unwise to demand long term (>1 day) stability of absolute lighting level, (as light intensity is bound to reduce with time, due to dust etc.), nor short term stability through stray reflections, such as due to white garments worn by humans etc. Therefore, having demanded constant lighting, the system must at least be insensitive to small variations in light level and direction. This is a principle that runs through the design of the vision system. The system demands certain conditions from the operating environment. However, having done so, it attempts to operate flexibly when that condition is not fully met i.e. it attempts to degrade gracefully as the operating conditions deteriorate from that required. The goal for coping with illumination variations was to make the system insensitive to as large a variation as possible. This could be achieved by 4 means.

1. By using a gradient image.
2. By using a learning algorithm to compensate for a wide variety of imaging and processing defects.
3. By using a variable edge threshold.
4. By using hardware help in the form of an auto-aperture lens.

The basic system reported in this thesis uses only the first two techniques. The use of the gradient makes the system independent of the

absolute illumination level within the limits set by the dynamic range of the input image and the need to threshold the gradient image to reduce the information content. The problem with the edge threshold may be removed by varying the edge detector threshold with the overall light intensity. However, the signal to noise ratio of the edge image deteriorates when this is done. This problem could be avoided by using an auto-aperture lens on the camera. The basic system (i.e. using only the first two techniques) is able to cope with fairly large variations in light intensity {section 5.2.1.1}. Variations of edge detector threshold could be used at the expense of extra processing (necessary to compute the threshold to be used). This allows the system to cope with larger variations in absolute intensity level. The use of an auto-aperture lens would make the system virtually insensitive to variations in overall illumination, limited only by the range of the auto-aperture lens and the sensitivity of the camera. The range of flexibility of the the software to illumination variations takes care of variations in aperture size of the auto-aperture lens from ideal.

Variations in lighting direction have the effect of changing the illumination level of local regions differently. This poses little problem to the system so long as the variation is within the bounds of absolute illumination variation for a minimum number of regions. (Note that when using an auto-aperture lens this will usually be true, within the range of the aperture, as the lens will open to allow average illumination of the scene

Illumination variations due to areas of shadow and highlight are handled in the same way. Provided that the shadow areas are

sufficiently illuminated (i.e. the measured intensity is not zero),† the system expects to pick up features in those areas. However, the system does not attempt to find and recognize or identify shadow edges for what they are i.e. it does not attempt to describe the lighting conditions in the scene, and is only interested in describing the scene in terms of known objects.

3.2.1.4. Scale variations (i.e. size variation parallel to P). The system imposes a condition of fixed scale on the industrial environment. This restriction is argued to be acceptable for two reasons.

1. It would probably be cheaper to provide a constant scale factor, i.e. by having a fixed camera at a fixed distance from the objects to be recognized, than to have a roving camera with complex control mechanisms. Therefore the capability to recognize objects at random scale factors may not be essential in industrial vision. (I have not had access to any market research that either confirms or contradicts this).
2. It is reported that even the human visual system may not be able to operate reliably under conditions of random scale variations.

However, having imposed the condition of fixed scale, the system attempts to be relatively insensitive to small variations in scale. In tests [section 5.2.1.2] it was able to cope with up to a 30% variation in scale.

† Severe shadowing is uncommon in conditions of 'reasonable' lighting (as defined earlier) as there usually are several light sources, which tend to 'fill' the shadows of each other.

3.2.1.5. 3D orientation variation (i.e. rotation of the object about an axis parallel to P). As indicated earlier, the vision system is limited to the recognition of stable states of objects. This constraint is commonly encountered in industrial vision systems. However, the system attempts to be insensitive to small variations in the 3D orientation of the object away from the learned plane. Ways of extending the system to cope with the full range of 3D orientation variation (i.e. 3D object recognition) is discussed in chapter {6}.

3.2.1.6. Partial Obscuration The system attempts to show insensitivity to partial obscuration of objects (i.e. due to 'overlapping' objects, or due to part of the object being outside the image frame). This is achieved by using local features of the objects.

3.2.1.7. Summary Thus there are 6 main imaging conditions to which the vision system is designed to show insensitivity in varying degrees. This insensitivity is achieved by using features that are themselves insensitive to these conditions. The learning algorithm {section 3.4} verifies this insensitivity. The chosen conditions are:

Complete range of values of
- 2D object position (within frame)
- 2D orientation

Large variations in
- Illumination
- (and hence small variations in lighting direction),

As large a variation as possible
- in object scale,
- 3D orientation, and
- partial obscuration.

The pre-processing begins with a gradient detection operation to allow maximum insensitivity to absolute illumination level. The gradient image is then processed to obtain local features. This processing is concerned with representing the gradient profile of local neighbourhoods by reducing the data content, while retaining the 'useful' information content, and at the same time, making the representation as independent as possible of the chosen imaging conditions. The pre-processing stage can be divided into 3 sub-stages as follows.

1. Gradient detection
2. Rep-point selection
3. Local neighbourhood selection.

I now describe the design philosophy, motivation, objectives, and justification of these processing stages. These descriptions should be seen as a specification of the processing required, and therefore attempt to be independent of the actual algorithms used.

3.2.2. Gradient Detection

As we have seen, the vision system begins processing with a gradient operation in order to reduce the sensitivity of the system to absolute lighting level. The scene and its objects are therefore modelled by their gradient profile. However, in order for local neighbourhoods to be represented and manipulated efficiently for matching and recognition, the data content of the gradient image must be reduced. This reduction must be achieved without reducing the 'useful' information content significantly.† This is possible because most

† The fact that there is redundancy in most images is seen quite clearly from research into image data compression, especially for low bandwidth picture transmission (e.g. Pratt [1978] Chapters 21-24).

images contain large areas of approximately uniform intensity. These regions transform to regions of approximately zero gradient in the gradient image. Regions with zero gradient can clearly be eliminated with no loss of information. This strategy can be extended by using a threshold to remove all areas of small gradient. The threshold is kept as low as possible to retain as much information as possible (for subsequent processing) within the available resources. This is a recurring principle throughout the vision system. Data is discarded only when the available hardware resources force us to do so. This allows the system to retain as much information as possible, and thereby respond to as many weak features as possible.

Thus, the gradient operation followed by a threshold operation results in an edge detection operation. An edge point, therefore, is defined to be any pixel with a local gradient greater than a given threshold. This allows the use of standard edge detectors such as Sobel or Roberts operators. The system, however, is required to be insensitive to the actual edge detector used i.e. the system operation must not depend on the use of a particular edge detector.

In addition to finding edge points, the edge detector is also required to compute the 'property' data of the edge points. The gradient direction of the edge point was chosen to be its property. Other parameters such as gradient magnitude, average local brightness, or average local colour may also be used. However, brightness data should not be used as it would increase the sensitivity of the system to the absolute illumination level. The edge detector is also expected to be able to cope with imperfections in the input grey scale image, such as those due to electronic noise.

Therefore, edge detection was used as the first processing step due to three reasons.

1. To reduce system sensitivity to absolute illumination level.
2. On the grounds of minimum information loss. (The gradient detection operation loses only the absolute illumination level as it is a differentiation operation. However, standard gradient detectors lose some high frequency information because of the use of a degree of local smoothing). This suggests a degree of reversibility of transform.
3. Easy control of the data content of the image by use of a threshold.

The following is expected from the chosen edge detector:

1. It should compute the edge property, and
2. be insensitive to noise in the image.

(An example of an edge detected image may be found in Fig. 5-3).

3.2.3. The Rep-Point algorithm

The output from the gradient operation is a list of edge points. This list usually contains from 2000-6000 edge points with the thresholds that are normally used.† This is a data reduction of over 90% from the original image containing 64k pixels. This list of edge points is still too large to be handled effectively for learning and recognition. (For example, it is not possible to consider each edge point to be a description of a small part of the object, and thereby

† These thresholds are not chosen dynamically, but are constant throughout the operation of the system. At present they are chosen empirically during the initial system set up process.

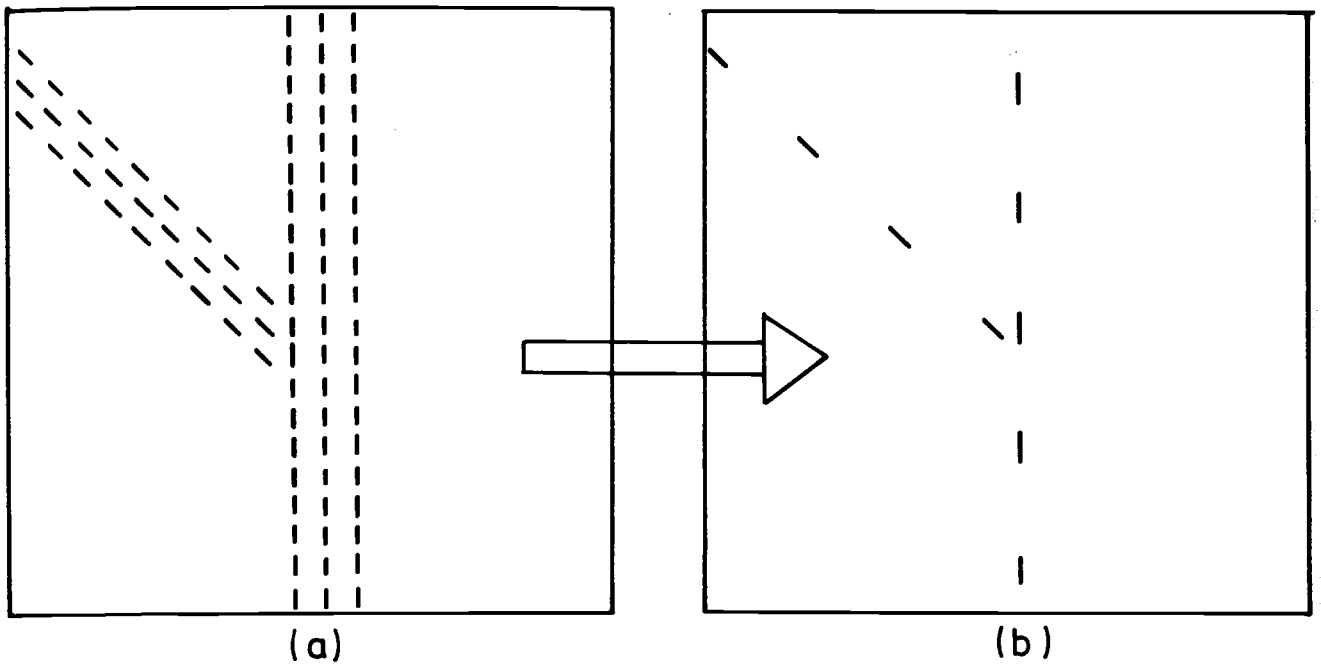


Fig. 3-1

use a relational structure of edge points. This would result in a graph of 4000 nodes!) The task of the rep-point algorithm is to reduce the data content further, without affecting the 'useful information content' in the image. This can be achieved by identifying 'redundancy' in the edge data. The acceptance of which data is redundant (in terms of the task at hand) could be a contentious issue. But, it is clear that once this is agreed upon, data reduction can be achieved without loss of useful information. My technique is to find representative points (rep-points) for small local areas of approximately uniform gradient property.

Fig. 3-1 to Fig. 3-4 illustrate the requirements placed on the rep-point algorithm. Given an edge image as in (a) of each figure, the rep-point algorithm is expected to generate an output as in (b) of the

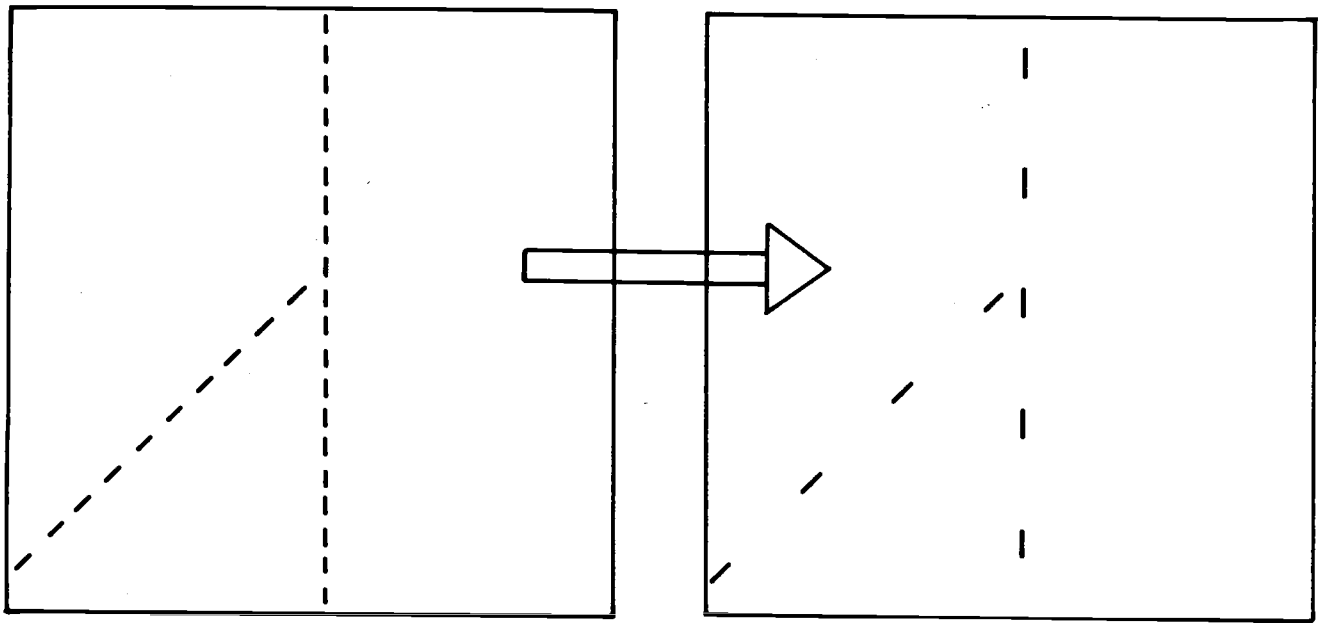


Fig. 3-2

same figure. The significance of these figures is described below.

Specification of the rep-point algorithm

1. Segment the thresholded gradient image (=edge image) into regions of connected edge points of approximately uniform gradient property. Region connectivity is 8-connectivity (i.e. two edge points are connected if the distance between them is equal to 1 or $\sqrt{2}$ pixels.)
2. Regions have an expected size of radius r. Regions of large radius should be segmented into several regions of smaller radius equal to about r. (see Fig. 3-1). Note that the radius of a region is interpreted quite liberally. The distance between the furthest points of the region may be used to define the radius. In general

regions are split only if they are larger than 4r.

3. The value of radius r should be chosen so that 2r>t, where t is the average expected thickness of edges in the image. However, r should be kept small (i.e. just larger than t/2) to keep information loss to a minimum.

4. A representative point (rep-point) is chosen for each such region so that the property of the rep-point is equal to the average property of the edge points it represents. The position of the rep-point is set equal to the mean position of the edge points. The motivation for using rep-point (and edge point) property is to make the rep-points as unique as possible to the gradient section being represented. For example, if property data such as local colour is used, it reduces the probability of a rep-point matching any other rep-point. In the implementation, however, only rep-point direction is used. This property is insufficient to stop a given rep-point from matching all other rep-points, but it reduces such matches to only one instance (i.e. a single orientation) per rep-point.

5. A set of uncorrelated edge points should be mapped to a similar set of uncorrelated rep-points.

6. Any rep-point that is nominated by only a single edge point is discarded as noise i.e. rep-points must represent two or more connected and correlated edge points. This improves the high frequency noise immunity. (see section {5.2.2.2})

7. The rep-point image is expected to be stable with line thickening and indeed with other variations. (Compare Fig. 3-1 and Fig. 3-2).

8. The rep-point algorithm is expected to represent complex gradient

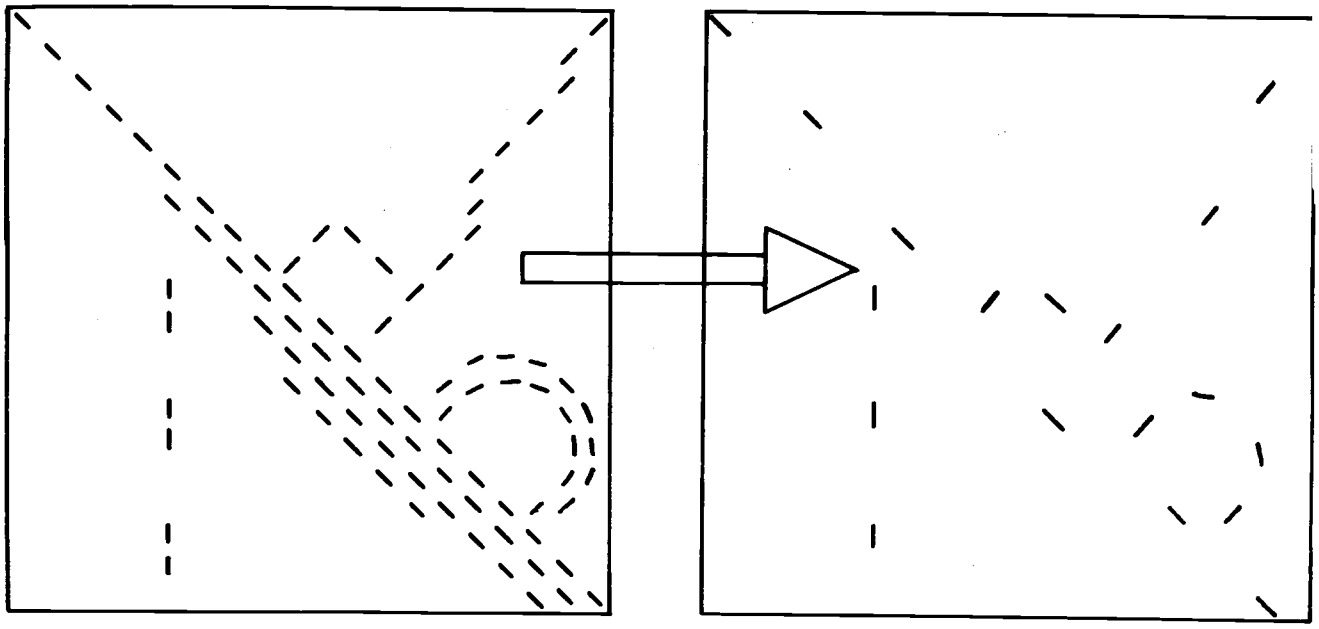


Fig. 3-3

profiles without restricting the allowed object geometry. (See Fig. 3-3).

9. The algorithm is expected to respond to weak features. (See Fig. 3-3).
10. Thin intensity bars should be preserved by using the gradient direction polarity. (Fig. 3-4).
11. Finally, when implementing this algorithm, (as indeed for any other pre-processing algorithm), a 'perfect' segmentation, although desirable, is not expected. Problems with implementing this algorithm may be passed on to the higher levels. (The matching and learning algorithms in this case).

Thus, the rep-point algorithm converts the edge image to a rep-point image. The algorithm copes with imperfections in the edge detector in two ways: (a) by averaging the property values of the seg-

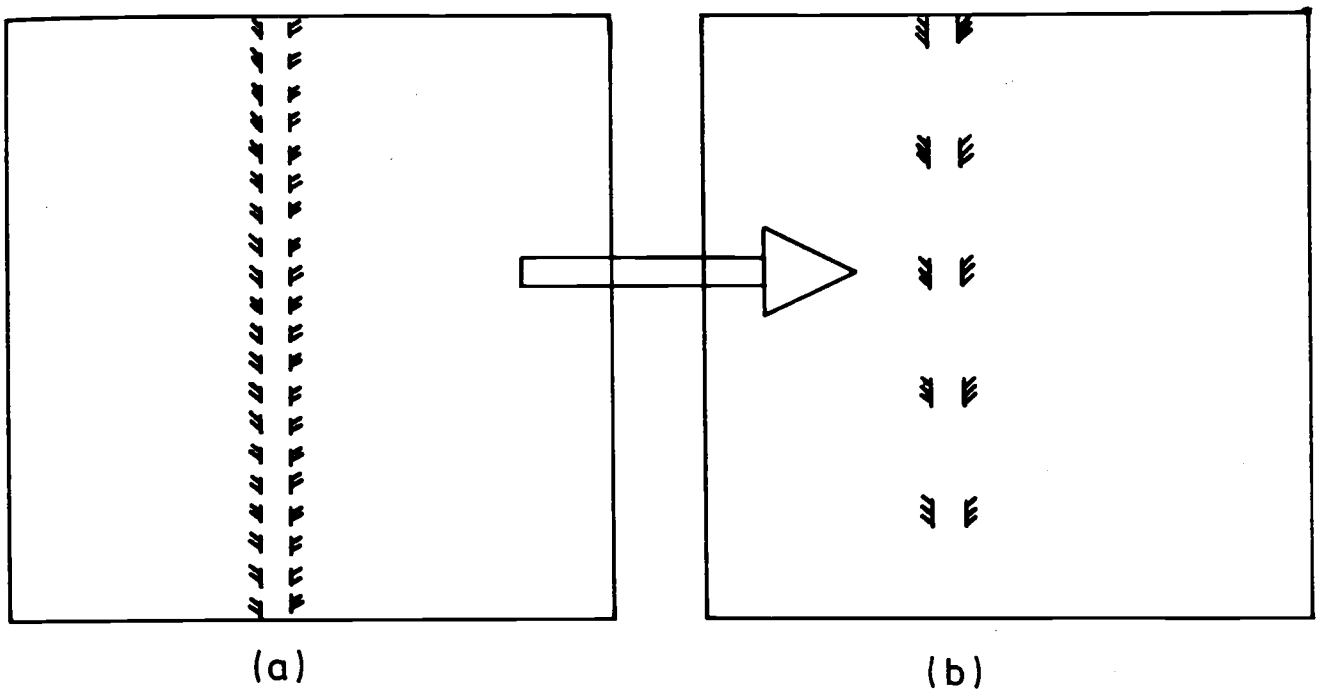


Fig. 3-4

mented edge regions, and (b) by requiring a minimum level of agreement between edge points (i.e. the requirement of 2 correlated and connected edge points to nominate a rep-point). This algorithm is similar to thinning algorithms [Hilditch 1969, Wong 1979, Athukorala 1980] but is different in that continuous lines are represented by a few rep-points. It is similar to region finding algorithms (if they are executed on a gradient image rather than a grey scale image), but is different in that large regions are segmented into smaller regions. It is similar to a simple resolution reduction algorithm, but is different in that the resolution reduction is a function of the gradient activity in the region. Areas of uniform gradient suffer a greater data reduction than areas of varying gradient. The number of rep-points generated increases with the spatial frequency of the gradient,

until adjacent edge points are judged to be uncorrelated.

Rep-points are the basic elements used by the vision system. Local gradient profiles are represented by rep-points. This scheme circumvents many problems of transforming edge points, such as thinning, chain coding (e.g. Freeman [1970], McKee and Aggarwal [1977]) and line finding (e.g. Shirai [1973], Mero [1981a]). I am not aware of an algorithm similar to the rep-point algorithm being used by previous researchers.

The rep-point algorithm has many advantages for my purpose.

1. The rep-point data is relatively reliable. As each rep-point is nominated by at least two correlated and connected edge points, it is less likely that the rep-point was generated by a random process. Further, the rep-point property data (including the rep-point position) is reliable as it is the mean of the edge points it represents.
2. It allows edge detector thresholds to be reduced without an accompanying explosion of rep-points. (See section {5.3.2}). This is because stronger edges ideally generate the same number of rep-points when the thresholds are reduced. However, the number of rep-points found for weak features will increase when the thresholds are lowered. If necessary, the strength of the edges may be used as an extra property, so that weak edges and strong edges do not form single regions. (i.e. rep-points will be either weak or strong). This differentiation is not made in my implementation.
3. The 'useful information' content of the image is expected to be preserved despite the data reduction. This can be justified on the grounds that uniform regions have a lower information content. The

final justification, however comes from the overall system tests in Chapter 5.

4. As explained in the next sub-section, the rep-points provide an ideal way of choosing local neighbourhoods. They also provide an ideal way of representing, manipulating, and comparing local gradient profiles {Section 3.3}.

5. No restrictions are placed on object geometry, as complex gradient profiles can be represented. Therefore, the system does not need to assume that the objects to be recognized contain straight lines or circular arcs. The complexity of geometry that can be represented is limited only by image resolution.

6. The last point leads to the possibility of representing textures. However, this is limited to textures that can be successfully represented by an edge image. This is effectively a requirement of minimum feature size of a texture. (The edge detector used may have to be changed to one using a smaller window size for this to be successful).

7. Finally, the algorithm is suitable for parallel processor architectures as serial algorithms are not necessary (unlike, for example, for line finding algorithms).† A cellular array processor would be well suited for this algorithm, although it was implemented in software with a pipelined architecture in mind. (See also chapter 6 on hardware implementation of the pre-processor).

(An example of a rep-point image may be found in Fig. 5-4).

† Although Hough transforms [Hough 1962] (which are parallel) may be used for finding lines and arcs, finding line termination points etc. can be problematic, and may need serial algorithms for efficient processing.

3.2.4. Constructing Local Neighbourhoods

The task of the final pre-processing stage is to select local neighbourhoods to describe the scene. Since local neighbourhoods are allowed to overlap each other, a very large number of neighbourhoods could be chosen over the image. Clearly though, local neighbourhoods chosen in regions of zero gradient activity (or sub-threshold gradient) will not be very informative. On the other hand, local neighbourhoods chosen in regions of significant gradient activity will be far more informative.

My technique is to use each rep-point as a focal point for selecting local neighbourhoods. This rep-point is called the central rep-point. This technique ensures that no neighbourhoods are chosen in regions with sub-threshold gradient (as such regions do not contain any rep-points), while a large number of neighbourhoods will be chosen in regions of significant gradient activity (which have a high density of rep-points). Therefore, the number of local neighbourhoods selected in the image will be equal to the number of rep-points. (Fig. 5-4, for example, contains 386 rep-points).

Specification of algorithm

1. Select a local neighbourhood of radius R around each rep-point. R should be chosen so that R>>r (where r is the expected average radius of a region represented by a rep-point). However, the value of R should not be too large in order to retain the locality of neighbourhoods. (I use a value of 9 pixel widths for R).
2. The gradient of the local neighbourhood is represented by the peripheral rep-points within the neighbourhood. Thus, local neighbour-

hoods are a relational structure of a single central rep-point and several peripheral rep-points. (Fig. 3-7).

3. Local neighbourhoods should be 'normalized' so that the rep-point data is represented relative to the central rep-point. (See Fig. 3-5). Local neighbourhood normalization results in an implicit rotation of the neighbourhood so that the central rep-point is oriented in an agreed direction. The significance of this is discussed later. {Section 3.3}.

The normalized local neighbourhoods produced by this algorithm are the features used by the system for learning and recognition. Local neighbourhoods, therefore, are also referred to as <u>local features</u> in this thesis, and should be understood to be synonymous.

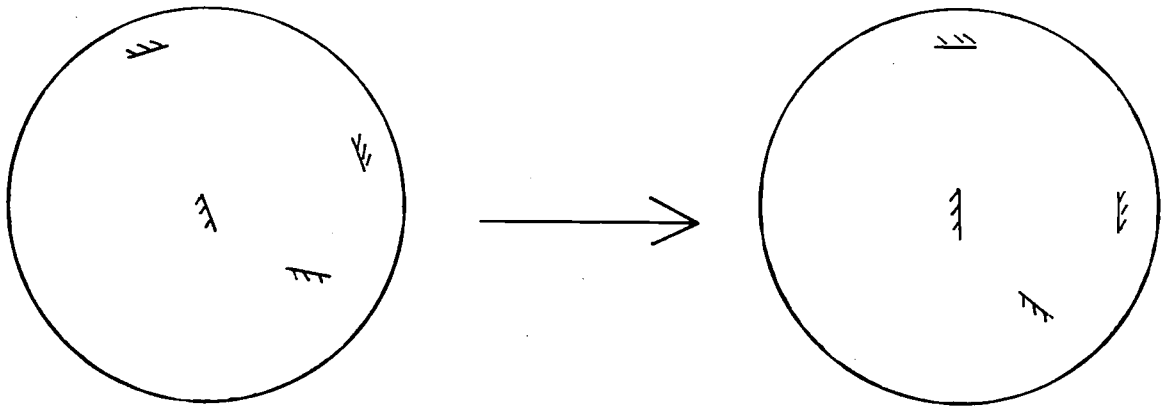


Fig. 3-5 Neighbourhood Normalization

Previous researchers have used local features for object recognition {sections 3.7.2}. However, they used 'conventional' features such as corners and holes. In addition to such features, my vision system is especially interested in 'unusual' local neighbourhoods (see Fig. 3-6) i.e. local neighbourhoods created by the juxtaposition of 'conventional' features. (In Fig. 3-6, the broken lines represent rep-points, and the unbroken line the boundary of a single feature. It is clear that such features are more complex than the local features, such as corners and straight lines, that have been used in the past). Such features tend to be more informative and unique, but have not been used in the past,† perhaps because of problems of building

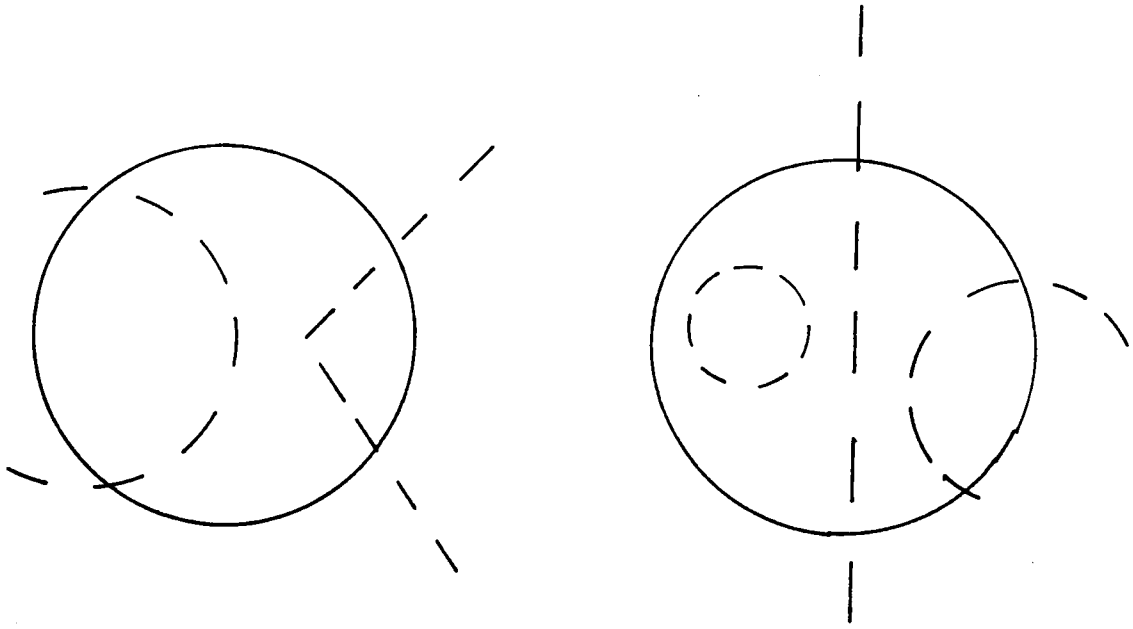


Fig. 3-6 The system is interested in unusual local features

† However, Persoon comes close. {section 3.7.2}.

feature detectors that could detect such features. The system is also designed to be sensitive to internal features of objects, (hence the attention given to weak features), including (reliable) surface markings and internal edges.

3.2.5. Summary of Pre-Processor

The task of the pre-processor is to extract a set of local features from the input grey scale image. Local features are overlapping local neighbourhoods of the gradient profile of the scene. The gradient profile is modelled by rep-points which represent small regions of approximately uniform gradient property. Regions with gradient below a chosen threshold are not considered.

Each local feature consists of a central rep-point and a set of peripheral rep-points. The rep-point data is normalized, i.e. represented relative to coordinate axis aligned with the central rep-point. This results in an implicit rotation of the neighbourhoods.

Local features, then, are able to describe complex gradient profiles. The technique of representing local neighbourhoods, together with the matching strategy, forms a generalized local feature detector. (cf. corner detectors, hole detectors, IC-pad detectors etc.)

3.3. The Feature Matching Algorithm

The matching algorithm is used by both the learning stage and the recognition stage to compare local features. The algorithm is expected to give a binary result of the comparison. Therefore, although the matching algorithm could be designed to give a value indicating the goodness of the match, only a binary result (i.e. match or not) is given. This is mainly because the learning algorithm and the recognition algorithm do not have a mechanism to handle partial feature matches. However, as any given local feature is one of many, and represents only a small local area of the scene, the loss of any one feature is not of significance. In contrast, the loss of a global feature (such as object area) could pose major problems to global feature based systems.

Thus, the task of the matching algorithm is to test for isomorphism of two relational structures. As seen in section {2.1.2.2} this is basically a graph isomorphism problem, but unfortunately there is no known algorithm that is both general and efficient. My feature matching algorithm exploits the special structure of the graphs (Fig. 3-7) to achieve an efficient match.

The matching algorithm operates by first superimposing (conceptually) the two neighbourhoods to be compared, so that the two central rep-points are aligned. It then counts the number of peripheral rep-points that coincide. The measure of coincidence is flexible so that small variations in the local features do not destroy the match. Two peripheral rep-points coincide if they have approximately the same orientation, and are less than a certain distance from each other. Two

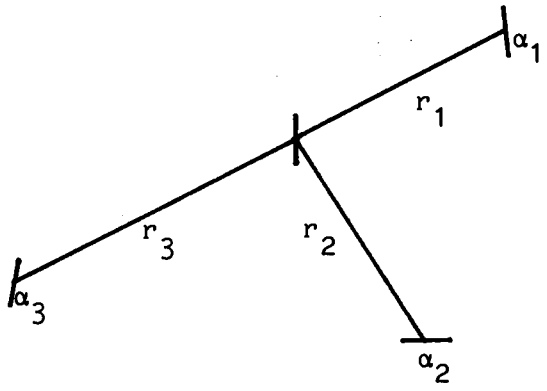


Fig. 3-7 Local neighbourhood relational structure with 3 peripheral rep-points.

neighbourhoods are matched if the fraction of peripheral rep-points that coincide is greater than a given threshold. The use of this threshold allows further flexibility of match. (However, in the implementation some of these thresholds have been removed {section 4.2}.) Thus, the matching algorithm compensates for imperfections in feature reproduction by the pre-processor in three ways.

(1) Features are matched even if they have a slightly different number of peripheral rep-points. In the implementation though, features must have an equal number of peripheral rep-points {section 4.2}. (Note that the number of rep-points in a local neighbourhood is a measure of the complexity of the gradient profile).

(2) The orientations of the peripheral rep-points are taken to be accurate, and result in a strict threshold being applied.

(3) However, the position of the peripheral rep-points are known to be variable, especially along their direction. A liberal threshold should be applied in the direction along the rep-point, while a stricter threshold may be applied in the direction perpendicular to the rep-point.

In the implemented system, the position threshold is liberal in all directions, and so is not dependent on the rep-point orientation. This has the advantage of simplicity and of allowing rep-points to expand and contract, and so allows a degree of scale change or distortion of the feature. The matching algorithm effectively tests for the angular relationship between rep-points, and the approximate position of the rep-points within the feature.

This algorithm executes rapidly for three reasons.

(1) Because central rep-points must be registered for a match to take place, they provide an ideal way of registering the neighbourhoods before matching. This means that two neighbourhoods can be registered in only one way, (because rep-point orientations are specified over a 360° angle), and therefore there is only one position in which two neighbourhoods can match. This eliminates the need to perform incremental relative rotations and multiple match attempts to verify a feature match.

(2) As all neighbourhoods have already been rotated by the normalizing algorithm, there is in fact no need for the matching algorithm to perform any rotations at all i.e. neighbourhood normalizing results in all neighbourhoods being already registered and ready for immediate comparison. This is the significance of the normalizing algorithm.

(3) It is possible to detect the non-matching condition of two dissimilar neighbourhoods very quickly. This is discussed in the implementation section.

But how can we be sure that two features that are matched by this algorithm in fact do correspond to the same object structure? There are two ways of ensuring this:

1. Firstly, the learning algorithm is responsible for ensuring that features are reliably matched. The algorithm observes the behaviour of features through variations in imaging conditions, and discards features that do not map to a single object. This is discussed in more detail in section {3.4}.
2. Secondly, it can be shown {appendix 1} that the probability of a match between two randomly chosen local features is small, and reduces rapidly as the radius of the local neighbourhood is increased. This is because the vocabulary of the feature descriptor is very large, and therefore the probability that two randomly chosen structures will be described by the same rep-point pattern is small.

Appendix 1 shows that the random match probability between two local neighbourhoods is small, under the assumption that all rep-point patterns are equally probable. However, this assumption does not hold for ordinary scenes. Certain rep-point patterns will be far more frequent than others, due to similarities in the object structure that is being viewed. Therefore, matches between rep-point patterns generated by similar object structure is more likely than matches for patterns generated by different object structure. But that is of course expected, and indeed required.

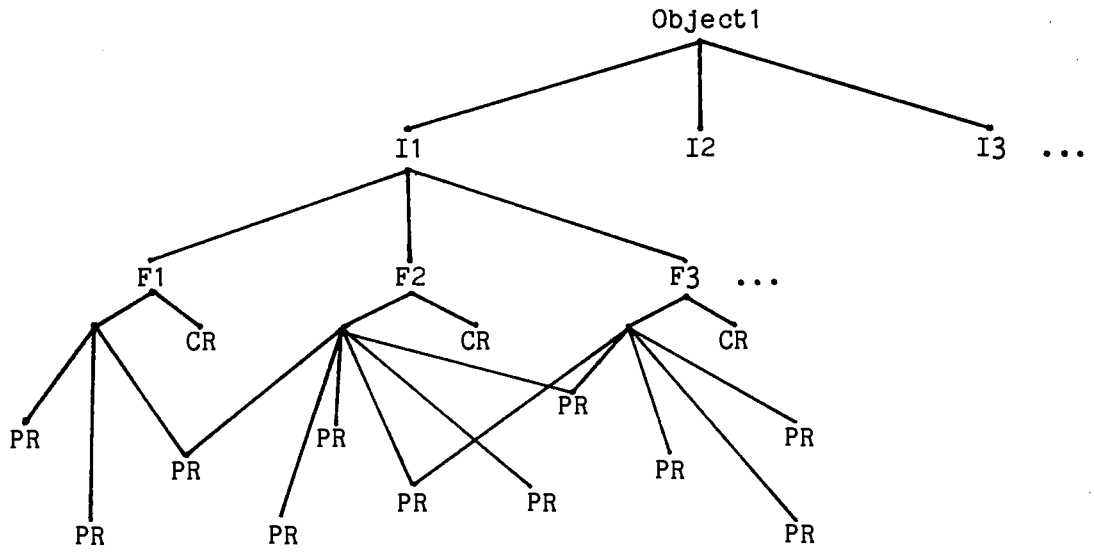
Thus the problem is as follows: When the system obtains a match between two features, it assumes that this was not due to a random event (especially when the number of rep-points in the feature is large). However, it cannot be certain that the matched features are not due to features taken from different objects that are similar to each other i.e. when a model feature matches an object feature, there is no guarantee that the new image feature was generated by exactly the same object feature. All that can be said is that the object structure in the vicinity of the feature is similar to the object structure that gave rise to the model feature. Therefore recognition of object features can be achieved only by learning about the way features are generated and changed. It is necessary to observe the generated features to see which features are most reliable in identifying the original object features through variations in imaging conditions. Thus, it is necessary to find a set of reliable features for recognition. This is the job of the learning algorithm.

3.4. The Learning Stage

This stage is divided into three parts.

1. Model formation to acquire an initial description of each object to be recognized.
2. Further learning by observing the reliability of the pre-processor over a set of imaging conditions for the objects of interest, by selecting a set of reliable and unique features.
3. Construction of a data structure for recognition.

3.4.1. Model Formation



I - Object Instances
F - local Features
CR - Central rep-point
PR - Peripheral rep-point

Fig. 3-8 The model data structure

During the learning phase, models are formed from each instance of the object that is taught to the system. This is effectively a storage of a feature description of the object similar to the learning stage of many previous vision systems. (e.g. Perkins {section 1.3.6})

Objects are described in terms of rep-points and local neighbourhoods. This data structure contains redundant data (because the local neighbourhoods can be generated from the rep-point data). This, however, allows faster execution of the learning algorithms. Fig. 3-8 shows the data structure of the models. The model data is a relational structure of features and rep-points.

3.4.2. The Extended Learning Stage

In chapter 2 it was seen that there was a basic need for further machine learning. Thus, the task of my learning algorithm is to learn by itself the way to use the features that are generated by the pre-processing stage i.e. the learning algorithm is required to examine the features, compare them, notice which of them are common, which are rare, which features are reliable, and so on. Thus, the learning algorithm is required to learn how to use the features detected by the pre-processing algorithms to recognize objects. The two basic requirements are (a) to verify the reliability (or independence) of the feature description through variations in the imaging conditions of interest, and (b) to select a set of unique features from this set of reliable features.

Previous vision systems operated by comparing image features with model features, and recognition was obtained if the feature sets were

sufficiently similar. This was done, however, without knowledge of the the importance of particular features in discriminating between objects. This resulted in difficulty in defining the measure of similarity especially when flexible operation was required. Some systems circumvented this problem by using human help to identify important features. (e.g. Yachida and Tsuji [1977], Mero [1981b], Tropf[1981], Rummel and Beutel [1982] etc.). The main objective of my learning algorithm is to find distinguishing features of objects automatically. The system aims to use any distinguishing features for this purpose i.e. it is not designed to be limited to features that the human programmer thinks is important. A feature is a distinguishing feature if it is reliable and unique to an object. Features such as those in Fig. 3-6 are especially important to the system.

Thus the learning algorithm attempts to answer the question 'What makes one object different from another object?'. The learning algorithm that has been implemented attempts to answer this question by using the local features of the objects (but not the relationships between the local features). The algorithm does this by finding a set of reliable and unique local features for each object. Chapter 6 discusses ways of extending this strategy to non-local structure and 3D interpretation. This section concentrates on the architecture of the implemented learning algorithm.

During the learning phase the vision system is taught several instances of each object placed in random (2D) positions and orientations. Reliability of features is computed by checking that a given feature is reproduced in all of the instances. This allows the system to reject

1. local neighbourhoods that were created by the coincidence of (say) shadow and highlight edges with reflectance edges,
2. local neighbourhoods that were disturbed by electronic or visual noise, and
3. local neighbourhoods that were disturbed by imperfections in pre-processing algorithms. This may happen due to threshold effects when choosing local neighbourhoods and due to imperfections in the implementation of the rep-point algorithm.

An important point here is that any feature that is reproduced through the chosen instances is taken to be reliable. But, some of these features may not correspond to a physical attribute of the object being viewed. For example, it could be created, (or modified), by stationary† highlights or shadows, or by imperfections in pre-processing algorithms. But this does not matter. A feature may be used for recognition as long as it is reproducible, even though it may be a result of an imperfect algorithm. This is an important principle. Vision guided action can be correct even if intermediate steps are in themselves judged to be imperfect or incorrect. Such imperfect features, or 'incorrect' descriptions of objects can be used to produce 'correct' actions. But then it is important to ask what is meant by a processing algorithm being incorrect, if the overall system functions correctly. It is clearly not necessary for a vision system to describe an object the same way as the human visual system in order for it to be judged to be a correct interpretation.

† Stationary with respect to object position and orientation (in the set of random views).

The pre-processing stage should be seen as performing an imperfect transformation of local neighbourhood gradient profiles. The higher processing levels can use this transformed output, nevertheless, by learning how the transformed output relates to the input image (i.e. by using the a priori knowledge in the learning stage that the sets of features generated for each object instance were formed from different 2D views of the same object). Therefore, imperfections in low level algorithms can be tolerated provided that changes in the input image produce changes in the transformed image over the parameters of interest; in this instance the parameter of interest is the actual object i.e. the task of the ideal pre-processor is to produce variation in feature description when different object structure is being observed, while keeping the description constant when the imaging conditions are changed. The task of the reliability algorithm in the learning stage is to observe departures from this ideal performance. Imperfections are compensated for by discarding features which are not reproduced reliably by the pre-processor.

Once a list of reliable features are formed for each object, this list is compared with all of the features found in the other objects. This is to see if a reliable feature found in object O_i has been found either as a reliable feature or as a spurious feature in instances of object O_j (for all $j \neq i$). All local features that are reliable and are not found in any other object instance form the list of unique local features. The number of unique features found depends on the radius of the local neighbourhoods and on the similarity between the learned objects. The number of unique features are expected to increase with the radius of the local neighbourhoods because each local

neighbourhood sees more of each object. See section {5.2.4.2} for empirical tests.

A major objective of the learning algorithm is to transfer as much processing as possible from the recognition stage to the learning stage. The recognition stage of a vision system spends much of its time searching. The objective was to move as much of the searching as possible to the learning stage. i.e., it was hoped that a richly connected data structure could be constructed at learning time so that the recognition algorithm had a minimum amount of searching to perform. The original aim was to achieve this by exhaustive comparison of model features so that an associative data structure could be simulated in software.

For example, if the recognition algorithm detected feature f1 in the image, the model data structure was expected to produce (1) a list of objects in which the feature could be found, and (2) the positions of f1 within the object. I expected to do this using object lists for each feature. An important (but I feel valid) assumption here is that the learning time is not critical.† However, it will be clear that this would require a large amount of storage, as features such as straight line segments would be detected everywhere, and would result in enormous object (and object position) lists. But this leads to an interesting point. The knowledge that a feature such as a straight

† It should be noted that since the learning algorithm is executed only once for a given set of objects, learning times of say 24 hrs even, are not disastrous. Further, if the learning time was critical, the learning algorithm could be executed on a larger machine, and the new data structures could be passed back to the smaller vision machine.

line segment is found everywhere is not a very useful piece of information for recognition i.e. long lists of objects are not very informative. They do not contribute much to the speeding up of the recognition algorithm. Clearly then, the long lists could be eliminated without a significant loss in the speed up of the recognition algorithm, but with a significant decrease in memory required to store the lists. My implementation is the limiting case of this, where only lists with a single atom is retained i.e. unique features.

The reliability test is an important part of the learning strategy. The original aim was to show each object in a large number of known positions and orientations and to make the system search for each feature in the position at which it should appear. In this way the system was expected to learn about the variation in rep-point patterns due to imaging condition variations. However, this would require the measurement of the positions and orientations of the object by hand, so that the system knew where the features were supposed to be, and could then compare this with the actual transformation of image features. This is unacceptable though, as it would require too much human intervention (and effort). Alternatively, the measurement and positioning could be achieved by a robot device. But, that would be an unacceptable requirement to place on an industrial user. Therefore, the present system operates without knowledge of the actual position and orientation of the object at each instance. During the reliability test, features in instance 1 of each object are searched for in the other instances. However, when a match is obtained the positions and orientations of the two matched features relative to the object are not compared. Therefore the reliability

test may confirm the reliability of a feature f1 by using a feature somewhere else on the object that matches f1. This is not altogether acceptable, but it is not a major problem either. This is because

1. it was shown in section {3.3} that the probability of a random match between features is small, and therefore any match obtained is due to similarity in the object structure and lighting conditions that gave rise to the feature, and
2. because any feature that matches other features easily will be rejected when inter-object comparisons are made. This is the case with 'common' features such as straight line segments which get discarded at an early stage.

In future implementations, however, it is expected that the position and orientation of matching features will be checked during the feature reliability test.

A useful side effect of the learning algorithm is that it allows objects to be learned on imperfect backgrounds i.e. features in the background will be rejected (so long) as they appear in at least two of the objects. In normal operation, though, it is expected that the system will learn objects on a featureless background so that the system has the least difficulty in deciding which features are object features and which are background features. (Note that in future systems which test for object feature positions and orientations in the reliability test, background features will be eliminated in any case as they do not move with the object). The system, then, does not place any constraints on the background used, except that a relatively featureless background would be desirable at learning time. However this is not necessary (see section {5.2.3.5}).

An extension to this idea is the possibility of using the system to recognize objects that are themselves variable from one instance to another. The reliability test can be used to extract the common features of such an object. Provided that there are sufficient common (unique) features, the system is able to recognize the object using these features.

Another useful feature of the learning algorithm is that it gives the user advanced warning of the expected performance of the system. For example, if the system detects a large number of unique features for each object, it will then be able to operate despite large variations in operating conditions. However, if it detects only a few unique features, the recognition will fail under smaller variations of the operating conditions. This scheme is superior to schemes where models of objects are stored, but not compared, so that any problems due to object similarity etc. are found only during the recognition stage. This means that these systems have to be tested on the number of correct classifications made out of, say, 100 trials. This is necessary because there is no measure available in advance of how different the objects are from the vision system's point of view. Therefore, my system cannot make mistakes <u>in principle</u> because it uses reliable unique features, and by definition the detection of a unique feature must imply the presence of the object. This is different from the standard strategy of computing the match weight of the input object with all of the stored models and then using an arbitrary threshold over global weighting criterion (e.g. the object is detected if 50% of the features are detected). However, my system could make mistakes if a unique feature is created by noise or coincidental

alignment of image features, and therefore, more than one unique feature is required to confirm recognition. But as seen in section {3.3} the probability of such a random event is low. I have not found this a problem in practice as it is rarely that more than 2 unique features are detected when they should not have been, even when the operating conditions were outside the required range, and dirt and swarf were present. However, this is dependent on the reliability tests during the learning stage. The feature reliability can be increased by extending the learning stage by using many more instances of each object.

Therefore, unlike in most previous vision systems in which the recognition algorithm was more complex than the learning algorithm, my recognition algorithm has been greatly simplified at the expense of an extended, and time consuming learning strategy. The advantage with this is that the recognition algorithm is able to execute rapidly.

The problem of recognizing objects using structural (relational) descriptions of objects is one of matching relational graphs. The strategy of previous workers has been either to use heuristic algorithms that take advantage of special characteristics of the particular problem {sections 3.7}, or the strategy of finding maximal cliques {section 2.1.2.2}. My proposal is to effectively eliminate the graph isomorphism problem in the recognition stage by searching only for unique relational structure. The graph isomorphism problem is then transferred to the learning stage, where the problem is magnified in scale i.e. the system is expected to find (in general) all unique relational sub structure of all combinations of features. This problem is simplified in my implementation by restricting the search to

local subgraphs of rep-points only. (Note that the time taken to compare 3 objects using 5 instances each to find unique local features is only about 3 minutes on a PDP11/24). Clearly, there is scope for looking for larger unique relational structures, especially as larger processors, more efficient algorithms, and longer execution times could be tolerated. Chapter {6} on architectural extensions discusses these possibilities in detail.

The effect of the learning algorithm can be thought of as being analogous to the effect of applying feedback† to an operational amplifier in analogue circuit design. Imperfect and noisy pre-processor algorithms are 'cleaned up' and made 'linear' so that only reliable and distinctive features emerge from the system. This is achieved by selecting the section of the pre-processor mapping function that produces a locally linear mapping between object structure and description.

3.4.3. Constructing a Data Structure for Recognition

Once a list of unique features is constructed for each object, the learning algorithm organizes this data for the recognition algorithm. The unique feature list for each object is first sorted so that the local features with the highest number of rep-points are at the head of the list. (i.e. features with the most complex gradient

† Applying feedback to an operational amplifier (op-amp) results in the non-linearities and distortion of an open loop amplifier being replaced by a linear response. However, this analogy does not extend to potential stability problems with feedback amplifiers as the learning system is not a feedback system. The analogy is only with the effect of feedback on op-amps.

profile are chosen). This is done for two reasons.

1. These features are least likely to be matched by chance as the random match probability drops rapidly when the number of rep-points in the neighbourhood increases. {appendix 1}.
2. Because local neighbourhoods with many rep-points are expected to contain more information than local neighbourhoods with fewer rep-points, as they represent complex gradient profiles.

The recognition data structure is a list of features that are to be searched for by the recognition algorithm in the input feature stream during the recognition phase. The feature list can therefore be organized so that the objects are searched for depth first, breadth first, or in some other mode (such as number of features being proportional to the probability of the object appearing). In the implementation, the recognition data structure is organized so that a breadth first search is performed. Therefore, the recognition feature list is

$$\{U_{11}, U_{12}, \ldots, U_{1n}, U_{21}, U_{22}, \ldots\}$$

where U_{jk} is the jth unique feature of the kth object.

This scheme allows the search strategy of the recognition algorithm to be changed by simply re-ordering this list.

3.5. The Recognition Stage

The task of the recognition algorithm is to search for unique features in the input feature stream from the pre-processor. The recognition algorithm depends on the knowledge that (by definition) the detection or the non-detection of a unique feature is a significant event. That is, the recognition algorithm depends on the following result that was derived in chapter 2:

$A_{k*} \subseteq FI$ when A_k is visible in the image, and

$A_{k*} \cap FI = \phi$ when A_k is not visible in the image.

That is, object A_k is recognized if any feature f is found such that $f \in FI$ and $f \in A_{k*}$. However, these equations hold only when a large number (i) of images are used for learning, and when these images are taken over a set of imaging conditions IC which includes the current imaging condition. However, if i is small, and we are not sure of whether the new image is taken with imaging conditions within the domain of IC, we cannot be sure that the above conditions hold, requiring more than one unique feature to be detected in order to confirm recognition. Thus, the recognition algorithm needs to be changed so that a small set of unique features F is found such that

$F \subseteq A_{k*}$ and $F \subseteq FI$.

The number of features required in F for a recognition to be declared, is dependent on what is known about the severity of the expected operating conditions during recognition. This is discussed in more detail in chapter 4.

3.6. Some General Points on the Architecture

3.6.1. Summary of Architecture

The architecture of the vision system is based on a new generalized feature detector and a learning algorithm. The bottom-up pre-processor is responsible for constructing a set of overlapping local neighbourhoods of the scene. These local neighbourhoods are the local features. The local neighbourhoods are represented by their gradient profile. The gradient profile is modelled by rep-points which represent small regions of approximately uniform non-zero gradient.

The task of the pre-processor is to generate a set of features that are independent of, or relatively insensitive to, variations in a set of chosen imaging parameters. The pre-processing stage consists of a gradient detection and thresholding operation, followed by a rep-point algorithm, neighbourhood selection and normalization. The neighbourhood selection algorithm is especially interested in regions of complex gradient profile.

The learning algorithm is responsible for learning to recognize the objects using the local features generated by the pre-processor. In order to do this, it first finds reliable features. It then uses the reliable features to find a set of unique features for each object. The recognition algorithm searches for these features in the output feature stream from the pre-processor. It identifies objects using the knowledge that the detection of a unique feature is a significant event.

The implementation shows that this scheme allows the system to cope with variations in operating conditions and to operate rapidly

under favourable conditions. Further, the system demonstrates that the visual world is very rich in local information which may be used to recognize objects. Indeed, it was not necessary to use intermediate or global features (described in chapter 6) for recognition as sufficient unique local features could usually be found. The test data in chapter 5 shows that the system was able to cope with large variations in operating conditions. For example, recognition was achieved despite a 70% reduction in light intensity, 30% reduction in scale, and 30°-40° variation in 3D orientation. Further, the recognition algorithm operated rapidly under favourable/good conditions (10ms-500ms).† This performance makes the system unique amongst reported industrial vision systems.

3.6.2. Another perspective of the architecture

This architecture can also be thought of as an extension of the generalized Hough transform [Ballard 1979]. Ballard (p.22) suggests that the generalized Hough transform could be extended by using pairs of edge points to reduce the complexity of the locus of the object origin. Increasing the number of edge points used increases the accuracy of the computed locus of the origin and decreases the freedom of movement of the origin, as the number of positions at which the chosen edge pattern can be found on the original object is reduced. In the limiting case the edge point pattern will be unique, and will result in a single point prediction for the locus of the origin. My system can be thought of as this special case of the generalized Hough

† Excludes pre-processing time (70s) which is expected to be reduced to a negligible level with the use of dedicated hardware.

strategy, except that the amount of data used in the computations is reduced by using rep-point patterns instead of edge point patterns.

3.6.3. 'Plane' Classification

How does the system fit into the classification of vision strategies into processing planes? {section 1.3.7}. The feature plane of my vision system consists of local neighbourhood data and rep-point data. The model plane is distinct from the feature plane in that it contains a learned description of the uniqueness of each object in addition to the models formed by using the local features and rep-points. Thus the unique features are the 'generated features' {section 1.3.7} that are searched for in the 'detected feature' list.

3.6.4. Suitability for Parallel Processing

The local nature of the processing necessary for recognition means that this architecture is inherently parallel. Therefore, this vision system is well suited for implementation on an array processor. Since all of the processing, including the recognition algorithm can be executed on such a processor, this architecture would make efficient use of the resources provided by an array processor. See chapter 6 for a discussion of the possible implementation of the pre-processor using dedicated hardware.

3.6.5. Limitations of the Vision System

This architecture, and the present implementation are limited in that

1. large variations of scale cannot be handled,
2. large variations in 3D orientation cannot be accommodated,
3. no attempt is made to cluster similar objects into classes, and
4. objects without unique local structure cannot be recognized.

The last of these is the most important of these limitations for industrial object recognition. This limitation may be overcome by extending the system to larger relational structures of the objects. Ways of extending the architecture to remove these limitations are described in detail in chapter 6.

3.7. Comparative Survey

This section attempts to compare the vision system with previously reported vision systems. However, I am unaware of any systems that report a similar performance, or systems that use equivalent algorithms. I survey comparable vision systems, which are of two kinds. Those that use straight lines and circular arcs (concurves) and those that use local features. I am mainly interested in systems that claim flexible operation, or noise tolerance, or the ability to recognize overlapping objects.

All of the following systems are limited to the recognition of stable states of objects. The descriptions within each section are chronologically ordered.

3.7.1. Systems based on concurves

These systems have an initial advantage over my vision system in that a degree of error correction is achieved by restricting objects in the scene to those that contain a significant amount of straight line and circular arc features. The error correction arises from the fact that a given line segment with noise is known to be a straight line and not, say, a wavy line. The error correction allows these systems to operate under fairly noisy conditions. The benefits from this error correction is analogous to the error correction achieved by digital representation over analogue representation of data in communication networks. The disadvantage with the scheme is that these systems cannot operate efficiently (or at all) when the objects do not contain large regions of straight lines or circular arcs. Although

these systems could attempt to operate by approximating the image features by small line segments and small arc sections, this removes the advantage of using concurve descriptions in terms of noise immunity and reduced processing requirements. Therefore, this is a major disadvantage of these systems. I believe that none of these systems would be able to operate efficiently or display the same degree of flexibility as my vision system if objects of the type in Fig. 5-1 were used. A further disadvantage with these systems is the need to perform line tracking in order to construct concurves. The tracking process can be seriously disturbed by texture and noise. Further, these algorithms are not inherently parallel, and therefore tend to be difficult to implement on parallel processors.

Mckee and Aggarwal [1977] report a system that uses binary images to form extended chain code descriptions of objects. (See section {1.3.1.1} for disadvantages of binary vision). Straight lines are fitted to the chain data. The system is able to recognize partial views of objects (not overlapping objects) by comparing generated line descriptions with stored descriptions. The comparison is made on subsets of the stored line descriptions so that small sections can be matched. However, two line descriptions are matched by comparing the area between their graphs normalized by the length of the comparison window. This therefore is a statistical test of matching (i.e. matching is not based on shape). The program is able to cope with scale variations, but the extent of variation allowed is not reported. The noise performance is not given either.

The system by Perkins [1978] (see description in section {1.3.6}) is reported to be able to operate in visually noisy scenes.

No figures are presented. The system is able to recognize partially overlapping objects, but no figures are given of the extent of overlap allowed. The system can handle only a 5% variation in scale. It is unable to cope with objects that contain textured surfaces. The recognition algorithm requires 0.1s-0.4s on an IBM 370/168 mainframe computer for simple scenes and 10s on scenes with multiple objects [Perkins 1977].

Mero [1981b] reports a system that describes objects using straight lines and circular arcs, and by using internal details such as holes. The system performs a heuristic search for these features. The search strategy for internal details is the same as that by Yachida and Tsuji [1977] {section 1.3.5}. The important features of objects are taught interactively (including the end points of the straight lines and arcs). The system is reported to be able to operate in noisy scenes. However, the system was tested using "shapes cut out from drawing paper". Therefore, the scenes were of relatively high contrast, and the objects had negligible height. The recognition algorithm executes in 0.5s on a VIDEOTRON R-10 minicomputer. (Preprocessing requires 5s on a 144x192 image). Results of the noise performance are not reported.

Presern and Kandus [1981] propose a system that uses concurves (i.e. straight lines and circular arcs). They perform a heuristic search, and the system is designed to be insensitive to noise. However, no results were given.

Dessimoz et al [1979] report a system that is able to recognize overlapping objects in noisy scenes. They use the Freeman chain code

which is then filtered and undersampled. Contours are compared by cross-correlation. Contour extraction is done using special purpose hardware. The processing time for tracking, filtering, and correlating was 10ms per contour point on a PDP11/40. (10s for 1000 points?) The extent of the noise immunity is not reported. The recognition is limited to the part on "the top of the pile" for overlapping objects.

Cheng and Huang [1981,1982] {section 2.1.2.2} use a sub relational structure called "star-structures" to match line segment descriptions of images. They use the system to identify aircraft in aerial images [1981] and to extract motion information [1982] in noisy scenes. The relational match algorithm is invariant through rotation, scale, and grey level modification. However the extent of noise etc. that can be handled is not reported. The processing time for matching a 70 node relational structure in the given example [1982] was 24.5s on a PDP11/70 minicomputer.

Hattich [1982] reports a system that uses line segments in grey scale data in a strategy similar to that of Tropf {section 3.7.2}. The system is able to recognize overlapping objects. Execution time, or level of operational flexibility is not reported.

Kimura et al [1982] report an algorithm for subpattern matching of line patterns (e.g. Japanese characters), through rubber sheet distortions. The algorithm is insensitive to scale changes, rotational variations, extra line segments, and does not use object dependent heuristics. The system is usable on industrial objects with user provided object models, objects being represented by straight line segments. Performance figures are not given for industrial objects.

3.7.2. Systems based on local features

The systems described in this section use relational structures of local features to recognize objects. All of the systems except the one by Persoon use 'conventional' local features such as corners and straight line segments.

Persoon's [1978/9] system has a similar motivation to mine in its use of local features. The system uses binary shape patterns of 11 pixel diameter with a frame size of 100x100. Objects are recognized by matching relational structures of "distinct" local shape patterns. A shape pattern is chosen if the centre of gravity of boundary points in the pattern is in the centre of the window (e.g. patterns with an edge running through the middle of the window). Distinct features are a list of features for each object, with each feature being different from the others within the list. This is different from unique features in that distinct features are unique within the same object, and not across objects. The local features used by Persoon are, however, not invariant through object rotation. Therefore each object has to be taught in "a large number of orientations in one quadrant". The distinct features found are then rotated by 90°, 180°, and 270°, and stored back in the distinct list as separate features. The features used are not invariant through illumination variations either, as binary images are used, and the features themselves are represented as binary patterns. (Note that the system therefore suffers from the other drawbacks of binary vision such as the inability to respond to internal features of objects, except for through-features such as holes, and the need for high contrast images.)

These local features are used to perform a heuristic search for local features in the input scene. This allows the system to recognize partially overlapping objects. The system is able to handle scale variations provided the features themselves are not affected by the scale change (i.e. the relational structure is scale independent, while the local features are not).

The processing speed, or the flexibility of the system with noise, or with variations in operating conditions is not reported. However, this scheme has clear advantages over standard binary vision techniques in that it can recognize overlapping objects, and is not sensitive to the loss of any given local feature as it does not depend on global statistics for recognition. However, compared with my vision system the drawbacks are as follows:

(a) The features are not rotationally invariant requiring the storage of, and the comparison with, different orientations of the same feature.

(b) The features have no built in resistance to illumination variations.

(c) The features have no built in resistance to scale variations and therefore to variations in object skew away from the learned plane.

(d) The system does not find unique structure of objects, and so the graph isomorphism problem is left to the recognition stage.

(e) Execution time is not reported by Persoon. However, I suspect that the execution time is not low because of (d) above, and because of the dependence on a relational match which is potentially explosive in processing requirements, especially if

arbitrary scale variations are checked for, and multiple objects are present in the scene.

Jacobus and Chien [1979] report a system that uses "half-chunks" to recognize objects. (See section {2.1.2.2} for a description). The system is independent of scale. Performance details are not given. However, Jacobus [1979] indicates that the low level processing requires 20 minutes of processing time, while the "graph-based manipulations" require 5 minutes per frame. The frame size was 252x238. The programs were written in Bliss10 on a DEC KI-10 processor.

Tropf [1981] reports a system that uses corners in grey scale image data to recognize overlapping objects. The object models are taught to the system by hand. During the recognition stage a heuristic search called "analysis by synthesis" is performed to tackle the graph matching problem. The system requires 0.5s for the analysis part (i.e. recognition time), while the pre-processing requires 30s on a SIEMENS 7760 computer. The noise performance or the extent of operational flexibility are not reported.

The system reported by Rummel and Beutel [1982] uses features such as "corners, straight lines, circles, grey levels and textures". Models are constructed from these features using human help to identify prominent features. The program performs a heuristic search to match image features to model features. The recognition routine executes in 210-640 ms for the first object in the image, running on a SIEMENS R30 minicomputer. The pre-processing requires 50s on a 128x128 image. The recognition time is "highly influenced by the selection of the first primitive in the model". An indication of the noise perfor-

mance or the degree of object obscuration allowed is not given.

Stockman et al [1982] report a system that uses local features such as line segments, curved edge segments, circles, and intersections. Registration is obtained by first matching image elements with all model elements on a local basis. Each local match produces a locus for rotation, scale and translation (RST) of the objects. However, the RST locus is reduced to a single point by using arbitrary combinations of two local features. (The combinatorics is controlled by using arbitrary rules for pairing points). The RST points are then clustered in a 4 dimensional space. This is a special case of the generalized Hough transform [Ballard 1979]. In their implementation the problem is simplified by assuming a fixed scale factor so that clustering needs to be done in 3 dimensional space. They use the system to register aerial images to maps, to detect airplanes in aerial images, and to recognize industrial parts. Object models are taught to the system by hand. The system was tested using "carburettor covers cut out of dark cardboard" and a set of (real) hinges. The system was tested on overlapping objects as well. Some difficulty in recognition is reported [p.239]. Execution times, and noise performance are not reported.

Bolles and Cain [1983] report a binary vision system that is able to use local features to recognize overlapping objects. The system uses 'conventional' features such as holes and corners. Objects are taught to the system interactively during the learning stage. The system then extracts a set of "focus features" for each object. A focus feature is a local feature used to focus the attention of the system during the recognition phase. When a focus feature is found, its neighbourhood is examined to detect other local features. Once the

matching local features are found, a graph matching technique similar to that used by Ambler et al {section 2.1.2.2} is used. Their clique finding algorithm, however, is different.

The aim of the focus feature finding algorithm is to find unusual, or unique local neighbourhoods around a chosen local feature. This is done on the basis of neighbourhood uniqueness over the given object set, and on the basis of uniqueness over the same object (i.e. a symmetry analysis). The feature selection process can be modified interactively. The motivation for this learning algorithm is similar to mine. However, as the model features are input by hand, there is no need for a reliability test. Further, the uniqueness tested is for extended neighbourhoods, which is similar to my intermediate features {section 6.1.1}. The local features themselves are limited to holes and corners, and therefore do not show any uniqueness. Therefore, the system depends on a relational match for recognition. This results in longer execution times. They report an execution time of 8s on a PDP11/34 to recognize 4 identical overlapping hinges when only the hinges were being searched for. Searching for 4 different objects takes 25s. (It is not clear whether this includes the time required to extract the local features from the binary image, but I think it does).

This system is limited in the present implementation to binary (high contrast) images.† Thus, these execution times have to be

† They claim that the system can be extended to grey scale images if the appropriate feature detectors were available. However, it must be stated that finding local features in low contrast images is a somewhat more difficult task, and that if the graph matching process had to take uncertain local feature matches into account, the effect on execution time could be significant.

compared with those for my system in favourable operating conditions (with multiple objects) due to their assumption of high contrast images. The performance of the system under noisy conditions, scale variations, etc. is not reported.

3.7.3. Comments on Comparisons

I am unaware of any vision system that reports a better processing speed coupled with the operating flexibility. Further, I am unaware of systems that use a similar strategy. However, Persoon had a similar motivation in choosing binary shape templates, and Bolles and Cain had a similar motivation in the feature selection strategy of their learning algorithm. I am unaware of any systems that have used a generalized local feature detector, where the local features were insensitive to a variety of imaging conditions and noise. I am also unaware of any systems that use unique structure to recognize objects. Finally, I am unaware of an industrial vision system that has a similar learning capability. Most of the systems described previously require user provided object models. The learning strategy of Perkins program is limited to a storage mechanism. Persoon's learning algorithm performs an extra degree of learning, by identifying a set of unrepeated features for each object. It does not attempt to formulate a recognition strategy, nor to test for feature reliability. The bulk of the processing is performed by the recognition algorithm. The learning algorithm by Bolles and Cain generates a recognition strategy, but the task is simplified by the absolute confidence available on the reliability of model features, as they are given to the system by hand, and by the use of binary images. Further, the results of the

learning algorithm are supervised by an operator to obtain optimal performance.

Chapter 4

Implementation: Algorithms and Data Structures

This chapter describes the implementation of the architecture described in Chapter 3. The algorithms are described in detail, and the design trade-offs are examined. All of the software was written in Fortran on a small minicomputer (PDP 11/24) running the RSX11M operating system. The grey scale images were taken from the vision system described in Athukorala and Wallace [1982]. The images were of 256x256 spatial resolution and 8 bits of grey scale resolution. Details of the user interface to this software is given in Appendix 2.

4.1. Pre-processing stage

Fig. 4-1 is a block diagram of the pre-processing stage. (A strategy for implementing the pre-processor using dedicated hardware is presented in chapter 6). The output of the pre-processing stage is used by both the learning and the recognition stages, and is a bottom-up process. Therefore, high level decisions do not affect the pre-processor. There were several design goals that had to be met when designing the pre-processor.

(1). Each algorithm had to be designed so that it would accomplish its task as best it could. However, no algorithm can expect a perfect

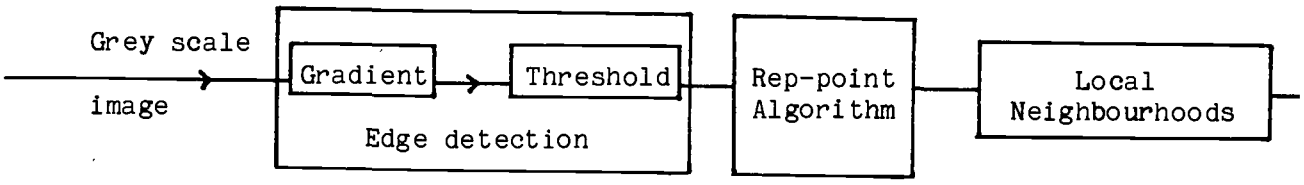


Fig. 4-1 Pre-processor

result from the algorithm preceding it i.e. imperfections in algorithm performance must be expected, and compensated for by the higher level algorithms. Therefore imperfections in the grey scale image (e.g. electronic noise) are compensated for by the gradient algorithm. Imperfections of the gradient image (e.g. edge orientation inaccuracy) must be anticipated by the rep-point algorithm. Imperfect segmentations by the rep-point algorithm (e.g. termination faults) must be handled by the matching algorithm. Imperfections in all of these is compensated for by the learning algorithm.

(2). A major objective when designing the pre-processing algorithms was to design them so that they could be implemented in hardware with minimum effort. I had the further objective of implementing the algorithms to execute reasonably fast on the PDP11/24. However, the latter was only a short term objective as I would not envisage the vision system being used without dedicated hardware.

(3). The pre-processing algorithms are expected to operate with fixed thresholds despite the wide variety of operating conditions expected. Threshold values could be tuned at system set-up time,

but they would then remain fixed during normal operation of the system. The test results reported in chapter 5, for example, have been obtained with a single set of thresholds.

The rest of this section examines the implementation of the 3 pre-processing stages: edge detection, rep-point selection, and local neighbourhood selection. Performance of these algorithms is discussed in Chapter 5.

4.1.1. Edge detection

There has been much research devoted to the problem of finding a high quality edge detector. Davis [1975] and Pratt [1978] p.478 survey the field. Raggett [1980] gives a number of references. See also Marr and Hildreth [1979], Abdou and Pratt [1979], and Beattie [1984]. I was not concerned with the problem of finding an ideal (or even a very good) edge detector. The system philosophy is to be able to cope with imperfections in algorithms as well as operating conditions. However, a few edge detectors were considered for the task. The 4x4 Walsh transform based edge detector (WTED) [O'Gorman 1978], the Sobel operator [Sobel 1970], the 2x2 WTED, and the Wong operator [Wong 1979] were considered. I was interested in identifying an 'efficient' edge detector, and the 4x4 WTED was chosen for the task. However, as reported in section {5.2.4.1}, the system was tested with other edge operators as well, and was found to operate satisfactorily.

4.1.1.1. The Walsh Transform based Edge Detector (WTED) This technique was first reported by O'Gorman [1978]. The WTED is similar in concept to the Hueckel edge detector [Hueckel 1973], but is based on

Walsh functions [Walsh 1923]. Walsh functions are a set of orthogonal functions $\{W_0, W_1, \ldots, W_k, \ldots\}$ which may be defined as follows.

When $k \geq 1$, $W_k(x) = W_d(2x)$ for $0 \leq x < \frac{1}{2}$

$= (-1)^{(k+1)} . W_d(2x-1)$ for $\frac{1}{2} \leq x < 1$

where $d = [\frac{k}{2}]$ and $F_0(x) = 1$ for $0 \leq x < 1$

([] means greatest integer less than.)

For example, the first 8 Walsh functions are shown in Fig. 4-2. The number of discontinuities in each function is equal to the order of the function. These functions are orthogonal.

i.e. $\int_0^1 W_k . W_j . dx$ $\begin{cases} = 0 & \text{if } k \neq j \\ = 1 & \text{if } k = j \end{cases}$

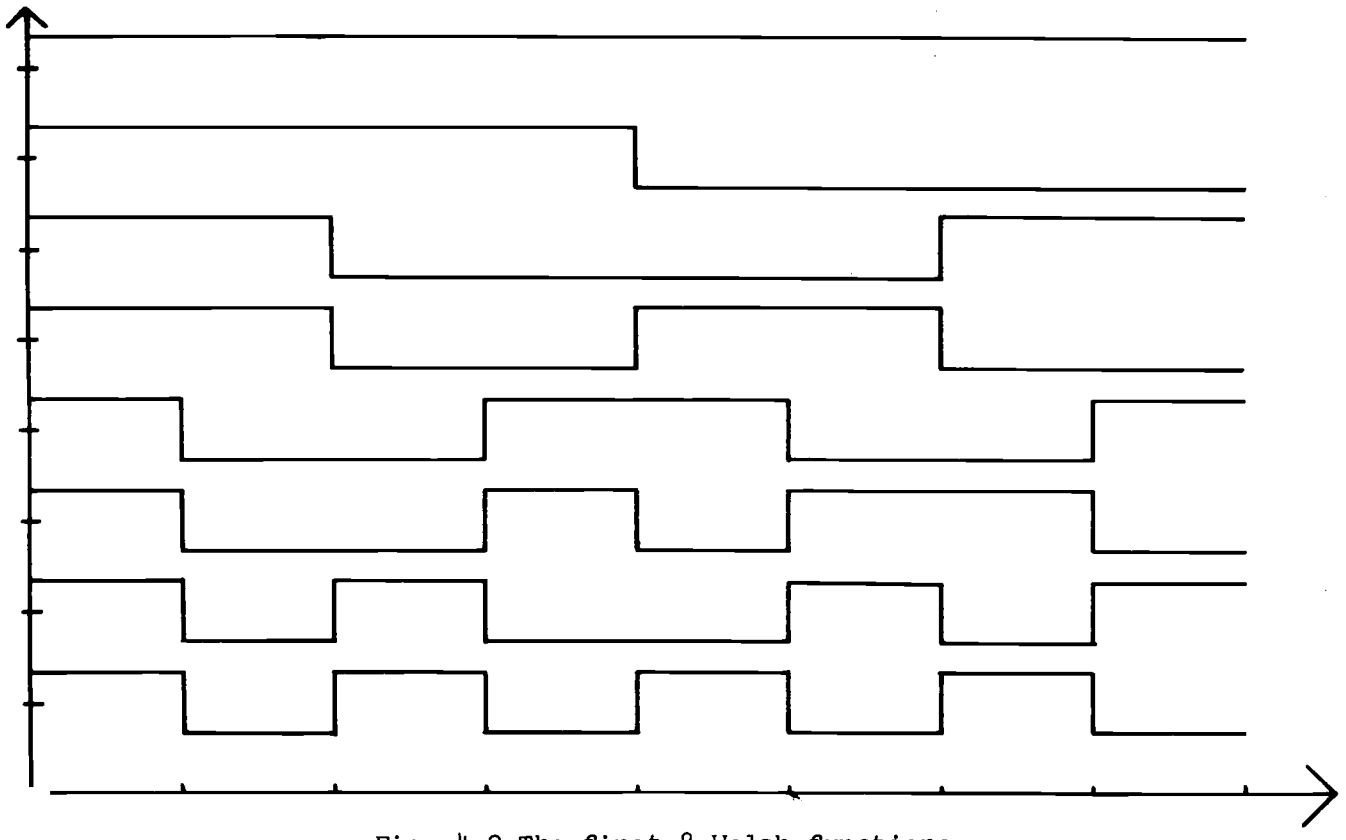


Fig. 4-2 The first 8 Walsh functions.

Therefore, a function $H(x)$ that is integrable on $0 \leq x < 1$ can be represented by an infinite series of weighted Walsh functions [Searle 1969]. (cf. Fourier series).

$$\text{i.e. } H(x) = \sum_{n=0}^{\infty} W_n . C_n$$

$$\text{where } C_n = \int_0^1 W_n . H(x) . dx$$

This can be done in two dimensions as well.

$$\text{Hence, } H(x,y) = \sum_{n=0}^{\infty} C_n . W_n(x,y)$$

$$\text{where, } C_n = \int_0^1 \int_0^1 H(x,y) . W_n(x,y) . dx . dy$$

Fig. 4-3 shows the first 16 Walsh functions in 2 dimensions. (Black squares represent -1's and the white squares represent +1's).

The strategy of the edge detector is to first represent the grey scale intensity of a 4x4 window, using Walsh functions. The advantage with Walsh function representation as opposed to a Fourier representation is (a) only 16 Walsh functions are needed to represent a 4x4 square precisely, and (b) only additions and subtractions are needed for the transformation. The weighting coefficients (C_n) are then compared with pre-computed weighting coefficients of a parametrized window with the required intensity profile. In the implementation, the parameters used are the step size, orientation, and the average brightness of either an ideal step edge or an intensity ramp (constant gradient) in the window. These parameters can then be computed from the coefficient comparison. It can be shown [O'Gorman 1978] that for a step edge or a constant gradient in the window, the following conditions are necessary.

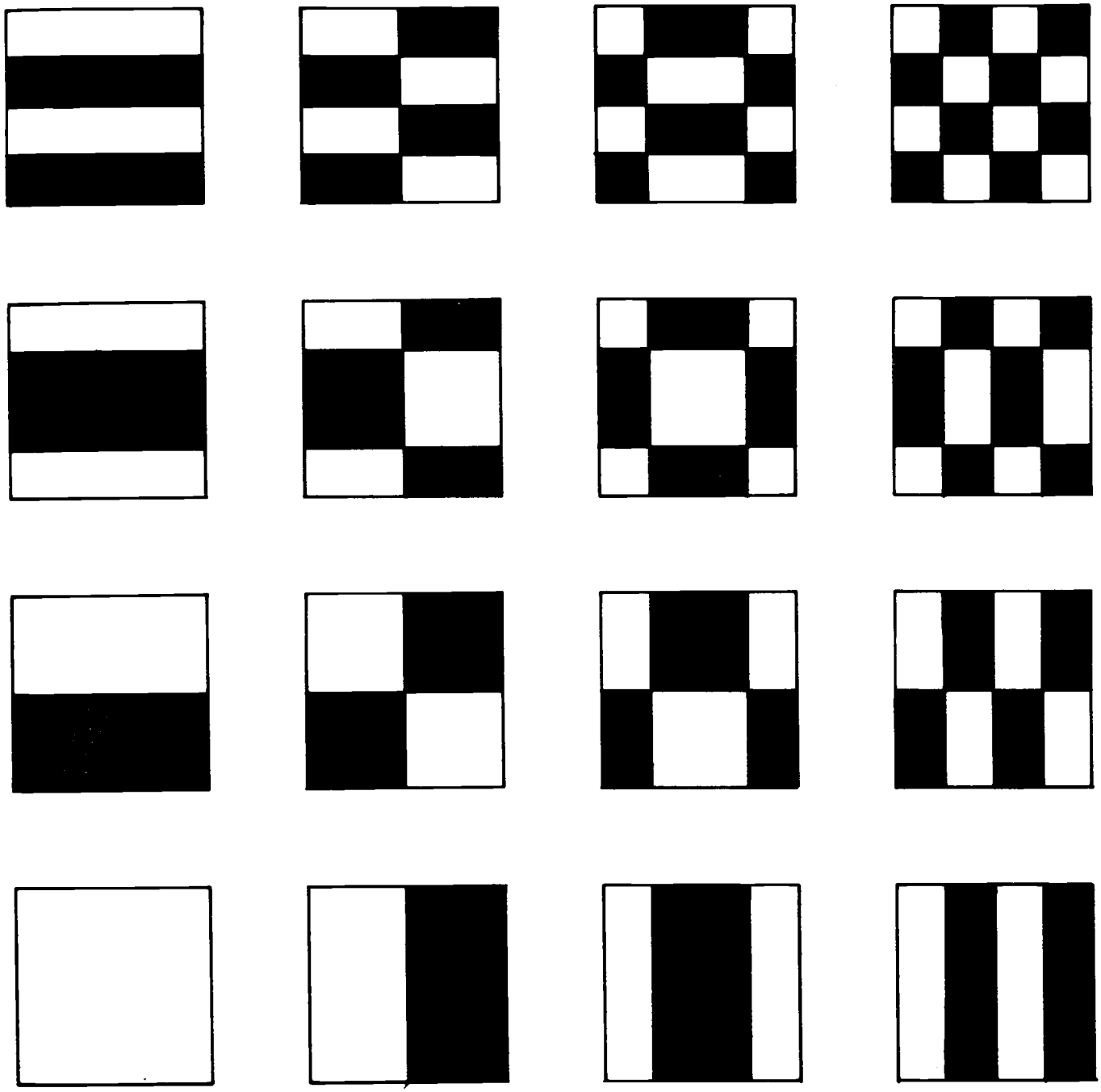


Fig. 4-3 The first 16 Walsh functions in 2D (Black=-1, White=+1)

$a_3=0$ $a_4=0$ $a_5=0$

where $\{a_0,\ldots,a_{15}\}$ are the 2D Walsh coefficients computed from the image window.

The 'goodness' of the match is tested using two thresholds, d and k.

$$d = |a_1| + |a_2|$$

and $$k = \frac{|a_1| + |a_2|}{|a_1| + |a_2| + |a_3| + |a_4| + |a_5|}$$

The gradient orientation is given by θ,

where $$\tan(\theta) = \frac{a_1}{a_2}$$

The expressions for k and θ are different from the those given by O'Gorman. The changes simplify the algorithms and allow the programs to execute faster. In this implementation, the WTED differs from a gradient detection algorithm only by the use of the k threshold. However, this threshold has an important effect on the performance of the edge detector in that it produces thinner edges. (See section {5.3.1} for comparison tests). However, the arguments in section {3.2.2} did not require thin edges from the edge detector. In fact, it may be argued that edge thinning is a loss of information. However, in order for the vision system to operate in a reasonable time, some data must be discarded. The advantage with the WTED is that it allows us to trade the information loss due to edge thinning with the information gain from reducing the edge detector thresholds (and thereby retaining weak features). This is because the WTED allows the 'd' threshold to be far lower than that of a gradient operator (such as Sobel) for the same number of detected edge points i.e., as the 'd' threshold is lowered, strong edges remain thinner with the WTED. This was one of the main reasons for choosing this edge detector.

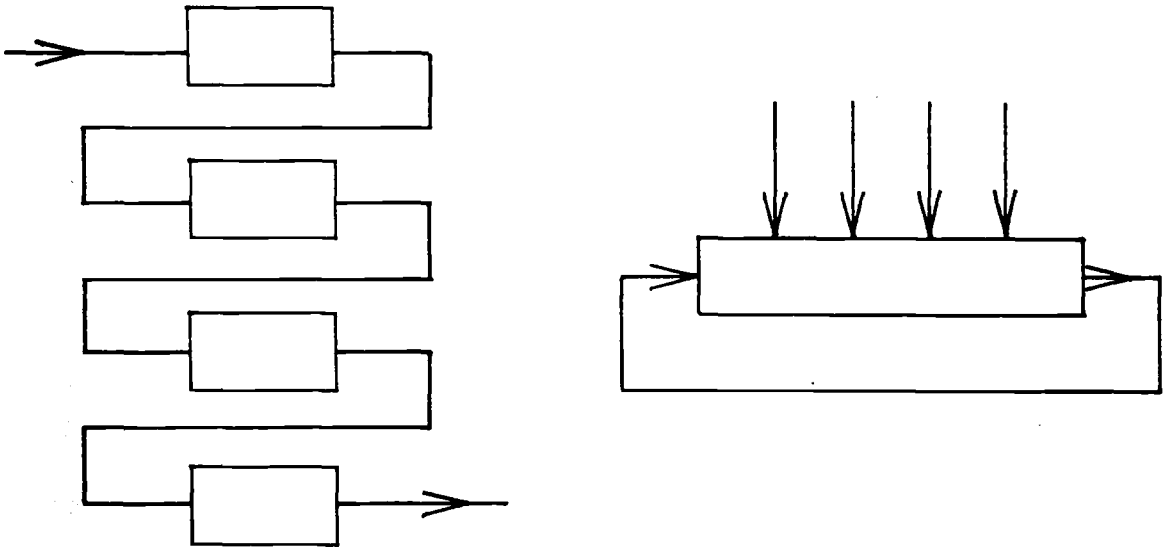
Another reason for using the WTED was its higher immunity to high frequency noise. This can be seen from the transform definition. The high frequency data is transformed into the higher order coefficients

(as in a Fourier transform) while it uses the lower order coefficients for the computation. Another way of looking at this is to consider the WTED as performing a certain amount of image smoothing (due to the 4x4 window) before the edge data is computed, resulting in a bandpass characteristic. The higher noise immunity of the edge detector coupled with the low noise, high quality image from the camera (quoted signal to noise ratio of 50dB) removed the need for an initial smoothing, or noise reducing stage. This saved a large amount of processing time (of the order of 50s per frame for the software simulation) and an extra stage of hardware processing when the pre-processor is implemented in dedicated hardware.

4.1.1.2. The WTED program The WTED can be implemented in software to execute reasonably fast, as multiplications and divisions are not required (except when computing k and θ. However, these two parameters are not computed for every pixel as the 'd' threshold is computed and tested for first. θ is computed only when an edge point is found - which is less than 10% of the time). Searle [1969] describes a fast Walsh transform algorithm (analogous to the fast Fourier transform) that allows the transform to be executed even faster.

The addressing restriction of the PDP11/24 made it necessary to store the grey scale image in virtual memory as a 32767 word linear array with two pixels packed into a single word. This made pixel addressing uncomfortable and slow, and required optimization of pixel addressing. If the algorithm was implemented without optimization, each pixel would have been accessed, and unpacked, 16 times. Optimization was achieved by using an integer ring buffer with four pointers.

The pointers were used to keep track of the window position within the ring buffer. Each virtual pixel was read and unpacked only once, and inserted into the ring buffer. Since the window pixels are accessed relative to the pointers, the ring buffer simulates a hardware serpentine memory. (See Fig. 4-4). The edge orientation is computed over the complete 360° range by using the direction of the intensity difference (i.e. the polarity). The computed edge orientation is then quantized to 256 levels (8 bits). The edge orientation was quantized to a resolution of 8 bits due to convenience of use, rather than due to expected accuracy. An 8 bit representation has the advantage of automatic wrap-around during angle arithmetic. The failure of the Fortran compiler to produce code to detect overflow and underflow conditions is thereby used to our advantage. The use of a high resolution representation also has the effect of minimizing quantizing noise.



Serpentine memory 'Ring' Buffer

Fig. 4-4

Inaccuracy of the angular data is anticipated by the use of soft thresholds (±30° for edge orientation, and ±18° for rep-point orientation).

The address of the edge point and its orientation are then passed to the rep-point algorithm. In the software implementation, the edge orientation is stored back in the original grey scale image, and the complete image is stored back on disc at the end. The stored edge image can then be used by the rep-point algorithm or by display programs.

4.1.2. The Rep-Point Algorithm

The implementation of this algorithm was expected to be the hardest of the three pre-processor algorithms. However, once the program had been coded and debugged, its performance exceeded expectations. Test results for the rep-point algorithm are given in section {5.3.2}.

The algorithm was implemented as two processes, and performs the segmentation in a single raster scan of the edge points. The first process segments the input edge data in the horizontal dimension to form 1D runs, and the second process collects vertically related 1D runs to form 2D rep-points.

4.1.2.1. The 1D rep-point algorithm This algorithm scans the input edge data in raster fashion, and segments the edge points into runs of similar edge orientation. Runs of length less than 2r are passed directly to the 2D algorithm. If the run is longer than 2r, the first r edge points are passed. The same criteria are used to segment the

remaining edge points. The algorithm can be stated as follows:

If $e_1, e_2, e_3, \ldots, e_n$ are consecutive edge points on an image row

and $|\text{Orientation}(e_i) - \text{Orientation}(e_1)| < \text{Orientation tolerance}$,

for any i,

then the run is defined as the set

$\{e_1, \ldots, e_n\}$ for $n<2r$,

$\{e_1, \ldots, e_r\}$ for $n\geq 2r$.

r is usually set to 4.

4.1.2.2. The 2D algorithm This algorithm attempts to perform the same process on runs in the vertical direction, as the 1D algorithm does on edge points. Rep-points are assembled in 256 accumulators (which is equal to the number of pixels on a row). Each new 1D run is then added to an accumulator. In order to do this, each accumulator is associated with a column of the image. All the accumulators that are near the mean column position of the new 1D run are tested to find the best accumulator for attaching. These tests are as follows.
(In the following, C_{1D} refers to the current 1D run, i.e. the new 1D run that is to be attached to an accumulator. O_{1D} refers to the 1D run that was attached last to the accumulator under consideration.)
An accumulator is chosen for adding C_{1D} to, if the accumulator passes 4 tests:

(a) The average orientation of C_{1D} is similar to the average orientation of O_{1D}.

(b) O_{1D} and C_{1D} are on consecutive image rows.

(c) C_{1D} is close to O_{1D}. (Horizontal distance must be less than R). The position of a run is the centre of gravity of the run.

(d) C_{1D} and O_{1D} are connected (8-connectivity).

These tests are designed to determine that all of the runs attached to an accumulator have similar orientation data, and that they are close together on the edge image. They also ensure that diagonal rep-points are properly formed.

If more than one accumulator passes all of the tests (which is not common), the 1D run is added to the accumulator in which the last 1D run is closest to the current 1D run. If the distance is the same, the 1D run is attached to the left hand accumulator (as the edge image is scanned from left to right). If an accumulator for attaching a new 1D run is not found, an empty accumulator is used to start a new 2D rep-point. Accumulator allocation is handled on a spatial basis (along the horizontal axis) so that accumulator searching is kept to a minimum. 256 accumulators are sufficient as long as the value of r is greater than about 2. Accumulators are cleared when a rep-point has been formed, and it is made available for new rep-points. Rep-points are formed from accumulated runs, as follows:

If $R_1, R_2, R_3, \ldots, R_n$ are runs in an accumulator

then the rep-point represents the set of runs

$\{R_1, \ldots, R_n\}$ if $n<2r$

$\{R_1, \ldots, R_r\}$ if $n\geq 2r$

At present, this section has been implemented as follows

$$\{r_1, \ldots, r_n\} \text{ if } n \leq R$$
$$\{r_1, \ldots, r_R\} \text{ if } n > R$$

This alteration allowed a simpler implementation of the algorithm but results in a slight asymmetry in the way the horizontal and vertical directions are handled.

4.1.3. Constructing Local Neighbourhoods

This algorithm may be implemented to execute 'on the fly' using a set of accumulators in a similar way to the 2D algorithm for rep-points. The accumulators would be used to hold unfinished features. Each input rep-point will be considered for attaching to all of the unfinished accumulators in which the central rep-point is close to the position of the new rep-point. Accumulators are freed when the vertical distance from input rep-points to the central rep-point is greater than the radius (R) of local neighbourhoods. (Note that local neighbourhoods are 'circular' i.e. the Euclidean distance between the central rep-point and the peripheral rep-points must be less than or equal to R. Therefore, neighbourhoods are circular within the bounds set by spatial quantization. See Fig. 4-5). However, the software implementation succumbed to the possibility of buffering the rep-point data. This allowed a simpler implementation.

Once local neighbourhoods are constructed, they have to be normalized. The normalizing algorithm is as follows.

Let the peripheral rep-points in the neighbourhood have

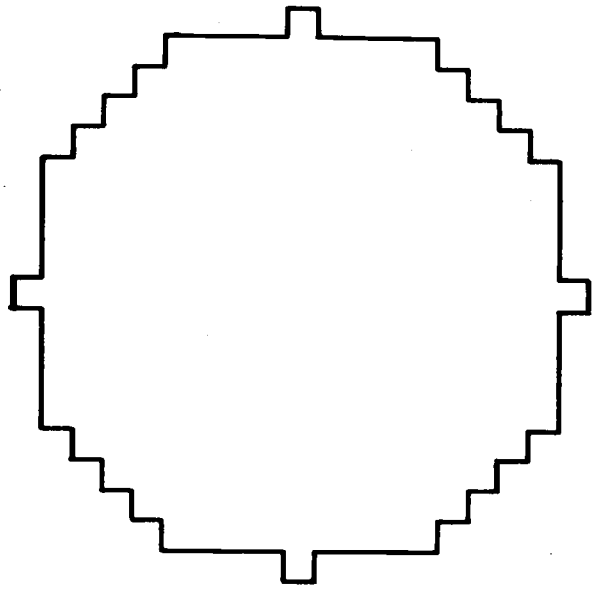


Fig. 4-5 Shape of a local neighbourhood of radius 9.

orientations given by $\{\alpha_1,\alpha_2,\ldots,\alpha_n\}$ relative to image coordinates, and positions $\{(r_1,\beta_1),\ldots,(r_n,\beta_n)\}$ where (r_i,β_i) are polar coordinates relative to the central rep-point. If the central rep-point has orientation α_0 relative to image coordinates, then the normalized neighbourhood will have rep-points $\{(r_1,\theta_1),\ldots,(r_n,\theta_n)\}$, with orientations $\{\gamma_1,\ldots,\gamma_n\}$, relative to the central rep-point,

where $\theta_i=\beta_i-\alpha_0$ and $\gamma_i=\alpha_i-\alpha_0$

The new cartesian coordinates (X_i,Y_i) of each rep-point relative to the central rep-point are given by

$$X_i=r_i.\cos(\theta_i)=r_i.\cos(\beta_i).\cos(\alpha_0)+r_i.\sin(\beta_i).\sin(\alpha_0)$$
$$=x_i.\cos(\alpha_0)+y_i.\sin(\alpha_0)$$

and $Y_i = y_i . \cos(\alpha_0) - x_i . \sin(\alpha_0)$

where (x_i, y_i) are the coordinates of the rep-points relative to the central rep-point before normalization.

In the implementation, the computation of sines and cosines, and of floating point multiplications, is avoided by using table lookup and integer arithmetic without a significant loss of precision, as follows:

As the angle values are quantized to 256 orientations, only 256 values of sines and cosines are needed. These can be represented by a floating point table of 256 values each. However, floating point arithmetic is time consuming, and unnecessary, as the final result is in fixed point integer representation. The sine and cosine values are therefore quantized to 513 values (from +256 to -256) with the understanding that +256 represents +1 and -256 represents -1. Therefore, the computation may be achieved by multiplying the integer coordinate value (x_i, say) by the integer sine value and dividing by 256. However, the division by 256 can be accomplished by simply choosing the high byte of the result (which is equivalent to a right shift by 8 positions). Therefore, computing $x_i . \sin(\alpha_0)$ which would normally require a sine computation, a floating point multiplication, and a real to integer conversion, is replaced by an integer table lookup and an integer multiplication. (This in fact can be achieved in a single machine instruction - i.e. when $\alpha 0$ and x are in registers- as the table lookup can be achieved with indexed addressing on the PDP11).

Local neighbourhoods are represented by a list of integers. Each integer specifies a single peripheral rep-point. The first byte of the integer gives the position of the rep-point relative to the central rep-point, and the second byte gives the orientation of the peripheral rep-point. This meant that the rep-points could have a maximum of 256 positions within the local neighbourhoods, which limits the maximum local neighbourhood radius to 9 pixels. (In fact, 9 pixel radius requires 253 codes). The codes are generated and decoded using two lookup tables.

```
    New code = XYtoCD(x,y)
X_coordinate = CDtoXY(code,1)
Y_coordinate = CDtoXY(code,2)
```

(Note: These arrays are vectored, so that no multiplication takes place during access. In the recognition algorithm, CDtoXY is equivalenced to 2 linear arrays CDtoX and CDtoY which eliminates a level of indirection for each access).

The implementation allows the system to operate with any neighbourhood radius of up to 9 pixels. However, the limit of 9 pixels is unacceptable, and should be removed in future implementations. Further, the requirement of unpacking data values using the lookup tables is also unacceptable. It is recommended that rep-point data values be represented using 3 bytes or 3 words depending on available memory resources.

One minor problem with this implementation is that some rep-points at the periphery of the local neighbourhood are moved outside

the local neighbourhood when the neighbourhood is rotated during normalization. This is due to the effects of quantization. At present these rep-points are simply discarded. A better strategy would be to move them in towards the central rep-point until they are inside the local neighbourhood again. However this problem will not arise in a new implementation that does not use a positional code.

4.2. The Matching Algorithm

The matching algorithm performs a flexible match between two neighbourhoods. It compares 3 parameters of the two neighbourhoods to be matched in the following order of precedence:

(a) The number of rep-points in each neighbourhood.

(b) The orientation of the peripheral rep-points.

(c) The position of the peripheral rep-points.

The general version of the algorithm is as follows:

Given two normalized neighbourhoods N1 and N2, where

$$N1=\{(p_0,\alpha_0),(p_1,\alpha_1), \ \ldots \ ,(p_n,\alpha_n)\}$$

and $$N2=\{(q_0,\beta_0),(q_1,\beta_1), \ \ldots \ ,(q_m,\beta_m)\}$$

where α_i and β_i are rep-point orientations and p_i and q_i are rep-point positions (including the central rep-points (p_0,α_0) and (q_0,β_0)), then N1 and N2 are matched if

$$\text{(a)} \ |n-m| \leq \text{THRESHOLD0}$$

$$\text{(b)} \ \sum_{i\neq 0} \frac{\text{match}(p_i,\alpha_i,N2)}{\min(n,m)} \geq \text{THRESHOLD1}$$

and

$$\text{(c)} \ \sum_{i\neq 0} \frac{\text{match}(p_j,\alpha_j,N1)}{\min(n,m)} \geq \text{THRESHOLD1}$$

$match(p_i,\alpha_i,N2)=1$

if there exists a (q_j,β_j) in N2

such that $|\alpha_i-\beta_i|<THRESHOLD2$

and $|p_i-q_j|<THRESHOLD3$

else, $match(p_i,\alpha_i,N2)=0$

(a) requires that the number of rep-points in the two neighbourhoods be similar, and (b) and (c) require that, if N2 is superimposed on N1, the percentage of rep-points in N1 and N2 which are similar in orientation and position relative to the their central rep-points be larger than a given threshold.

The above requirements have been designed to cope with missing rep-points and degradation in the input data. The matching algorithm was found to be relatively insensitive to THRESHOLD0 and THRESHOLD1. THRESHOLD0 was therefore set to 0, and THRESHOLD1 was set to 100%. It was now possible to rewrite the algorithm, resulting in a significant increase in execution speed. The new criteria for matching N1 and N2 are

(a) $n=m$

(b) $\sum_{i\neq 0} match(p_i,\alpha_i,N2)=n$

and (c) $\sum_{i\neq 0} match(p_j,\alpha_j,N1)=n$

Since the sum in (b) or (c) cannot be greater than n, the non-matching situation is easily detected (i.e. $match(p_k,\alpha_k,N)=0$ for any k). This contributes to the increase in execution speed. An important feature of the matching algorithm is the movement allowed for rep-points within the local neighbourhood. This enables the algorithm to match

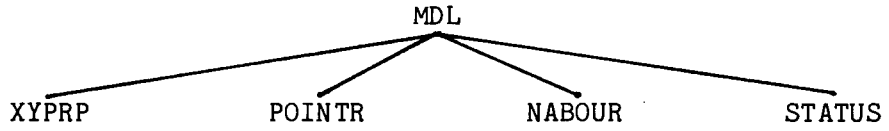
local neighbourhoods that have been distorted, for example, by scale variations. The function MATCH has the feature of allowing a single rep-point to be mapped to more than one rep-point in the corresponding local neighbourhood which results in further matching flexibility.

4.3. The learning stage

4.3.1. Model Formation

Each object instance that is taught to the system is stored as a model on disc. A model directory is maintained so that the software can keep track of the objects that have been taught, and what processing has been done.

The data structure used is of fixed size, which allows rapid access of the data. The model data structure consists of four sub structures as follows:



XYPRP contains the rep-point data. It can store up to 600 rep-points, and contains the X and Y coordinates of each rep-point and their property.

NABOUR is a list of neighbourhoods, which can store up to 2500 normalized peripheral rep-points. POINTR is a pointer array that points to the location of feature lists within NABOUR. POINTR contains the starting position of each list and the length of the list. This organization (see Fig. 4-6) has been chosen to allow rapid processing

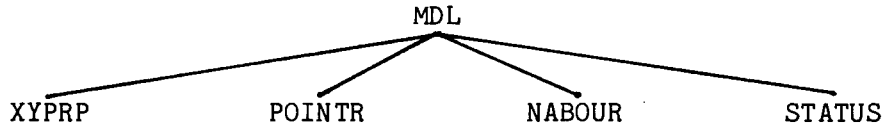
local neighbourhoods that have been distorted, for example, by scale variations. The function MATCH has the feature of allowing a single rep-point to be mapped to more than one rep-point in the corresponding local neighbourhood which results in further matching flexibility.

4.3. The learning stage

4.3.1. Model Formation

Each object instance that is taught to the system is stored as a model on disc. A model directory is maintained so that the software can keep track of the objects that have been taught, and what processing has been done.

The data structure used is of fixed size, which allows rapid access of the data. The model data structure consists of four sub structures as follows:



XYPRP contains the rep-point data. It can store up to 600 rep-points, and contains the X and Y coordinates of each rep-point and their property.

NABOUR is a list of neighbourhoods, which can store up to 2500 normalized peripheral rep-points. POINTR is a pointer array that points to the location of feature lists within NABOUR. POINTR contains the starting position of each list and the length of the list. This organization (see Fig. 4-6) has been chosen to allow rapid processing

of the data structures. Note that this pointer structure allows random access to the feature lists (i.e. the NABOUR lists do not have to be searched sequentially).

STATUS contains 128 words of status information. The status information records all of the processing done on the given image, the algorithms used, the thresholds used, and the result of the processing, such as the number of edge points found etc. The status information also serves as a check to stop incorrect sequences of algorithms being used. Such checks are invaluable during program development.

The complete model structure is stored in a contiguous memory block and is 'equivalenced' (Fortran EQUIVALENCE) to a linear array named MDL. Therefore, the model may be referenced as a single struc-

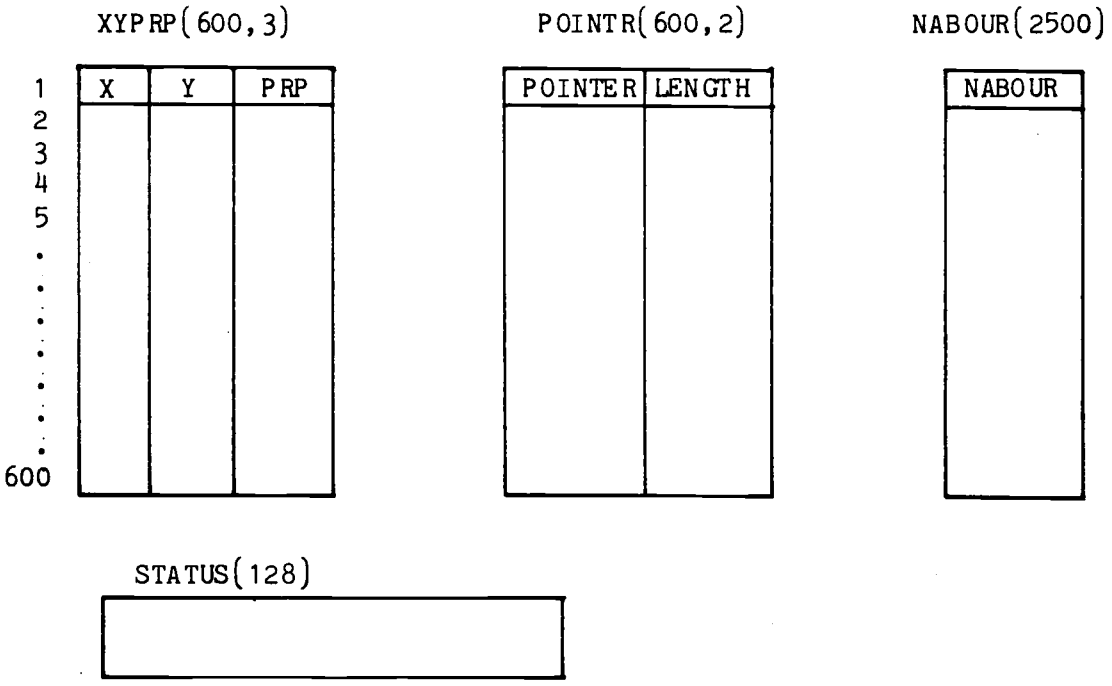
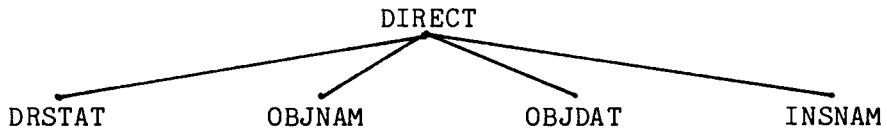


Fig. 4-6 Model data structure

ture. This allows models to be swapped in and out of memory quickly. In fact, the model data is read and written to disc using block I/O which executes rapidly.† The model data structure requires 5628 words of store.

The model directory is used to keep track of the objects known to the system. It contains information regarding the names of objects, the disc file names of the object instance models, and the processing carried out on the models. (i.e. what learning has taken place). The model directory structure is as follows.



DRSTAT stores status data of the overall system. OBJNAM remembers the names given to the objects. OBJDAT contains additional status data on each individual object. INSNAM remembers the names of the individual disc files that store the object models. This data is used by the learning algorithm during the learning phase to automatically access the correct files containing the object models. The object names are used by the recognition routine to report recognition success. The model directory structure requires 505 words of store, and is also read and written to disc rapidly using block I/O.

† Block I/O does not require any formatting, and does not use an intermediate data buffer. Therefore, it executes with minimum CPU intervention. (Data is moved to disc by direct memory access - DMA).

4.3.2. The Learning Algorithm

The learning algorithm is executed after all of the objects have been shown to the system and all of the models have been constructed. The algorithm first forms a set of common features (C_i) that are common to each object across the set of object instances (i.e. C_i contains the list of reliable features). C_i is then compared with all of the instances of the other objects so that all matching features in C_i are deleted. The remaining features are unique to object i.

The algorithm is as follows:

Let the object set be OBJ, the instance set be INST, and the neighbourhood set be NBHOOD. Then each object $O \in$ OBJ has a set of image instances $I \in$ INST, and each instance $i \in I$ has a set of neighbourhoods $N \in$ NBHOOD. In the following, I will use the operator '.' to select elements of a set. For example, $O_i.I_j.N_k$ refers to the N_kth neighbourhood of the I_jth instance of the O_ith object. (see Fig. 4-7)

(a) Then, for each O_i, construct a set of common neighbourhoods C_i, such that each neighbourhood $C_i.N_j \in O_i.I_1$, and there exists $O_i.I_m.N_k$ for all $m \neq 1$ such that N_j and N_k are matched.

(b) For each O_i, construct a set $U2_i$ of unique sets, such that each $U_j \in U2_i$ is unique to O_i with respect to O_j, $j \neq i$ i.e. each neighbourhood $U2_i.U_j.N_k \in C_i$ and there does not exist an $O_j.I_m.N_p$ for all m such that N_k and N_p are matched.

(c) For each O_i, construct a set UG_i of unique neighbour-

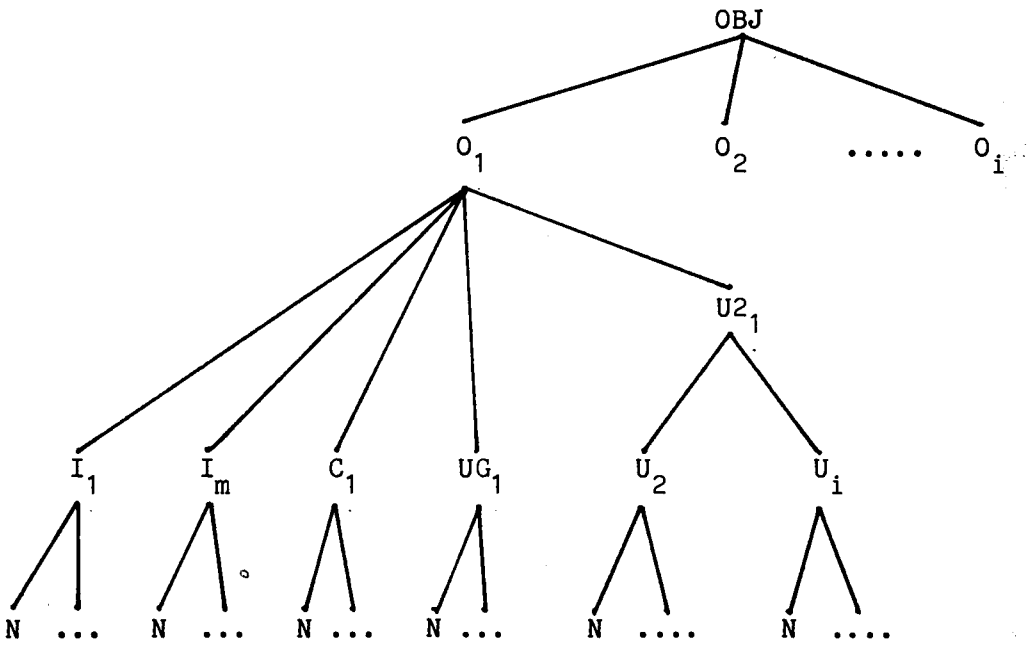


Fig. 4-7 Data structure used by the learning algorithm

hoods, unique to object O_i with respect to all of the other objects.

That is, $UG_i.N_k \in U2_i.U_j$ for all $j \neq i$.

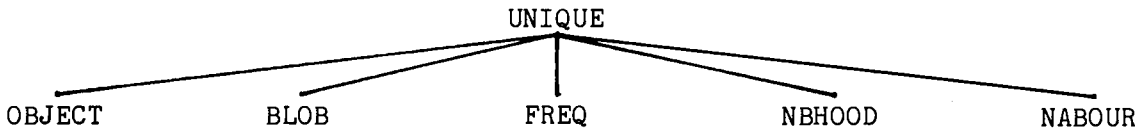
The intermediate sets U2 in (b), (which describe the differences between any two objects), are computed as it allows the system to be extended in future by clustering similar objects into classes (see section {6.1.4}).

The software, at present, allows 5 image instances per object, and up to 10 objects in the object library. The comparison routine takes approximately 2-5 minutes for 3 objects. The time taken

depends on the number of neighbourhoods found in each object, (and therefore on the size of the object). In the worst case, the execution time for learning is proportional to the square of the number of objects . However, in practice the program executes much faster due to the neighbourhoods that get discarded, and because each neighbourhood is compared only with a fraction of the other neighbourhoods due to variations in neighbourhood sizes. The algorithm can be speeded up for incremental learning by saving the results of the intermediate sets U2 in (b) above.

4.3.3. Recognition Data Structure

The recognition data structure is similar to the model data structure, and contains the following information:



OBJECT(i) stores the object number of the ith unique feature. BLOB(i) remembers the rep-point number of the feature within the first instance of the object during the learning phase. FREQ(i) indicates the frequency with which the feature has been detected by the learning or recognition algorithms. NBHOOD(i) stores the size of each feature in terms of the storage required in the NABOUR list. The NABOUR list contains the normalized feature data.

In the present implementation each object is allowed a maximum of 32 unique features. (The object on the left in Fig. 5-1, for example, had 51 unique features when compared with the other objects in the

same figure).

The unique data structure too is equivalenced and can be addressed as a single unit of 3828 words. It is read to and written from disc rapidly using block I/O.

4.4. The Recognition Algorithm

It was shown in section {3.5} that the recognition algorithm depends on the strategy worked out in chapter 2 i.e. an object is recognized if a single unique feature is recognized. However, this depends on the learning algorithm being executed properly i.e. the learning should have taken place with a large number of sample images. Since we have reduced the number of sample images used to learn from, our confidence in the identity

Object A_k recognized if $f \in FI \wedge f \in A_{k*}$

is lowered. Thus it is necessary to detect more than one unique feature to confirm recognition. Emperically it was found that the number of unique features required for recognition was dependent on whether the imaging conditions used during recognition were within that assumed by the system, so that a single unique feature was sufficient for recognition. It was also found that about 6 unique features were necessary to confirm recognition when the operating conditions were well outside the assumed conditions. The fact that the system operates at all outside the assumed conditions demonstrates the robustness of the strategy.

It is possible to optimize the recognition algorithm depending on the a priori knowledge of the kind of operating conditions that may be

encountered. On this basis the operating environment can be divided into 3 categories.

1. Favourable conditions: This is defined as the best operating conditions that the system could ask for, such as high contrast images (by using a black background for instance), diffuse lighting conditions, etc.
2. Good operating conditions: This refers to the conditions specified by the vision system as being necessary i.e. reasonable lighting conditions, fixed scale, and views of stable states of objects.
3. Poor operating conditions: This refers to operating conditions that fail to meet the required operating conditions i.e. when the operating conditions are not <u>good</u> as defined above.

The method of presenting objects to the vision system can be divided into 4 modes in order of expected difficulty.

1. Single object in image.
2. Multiple objects - not touching.
3. Touching objects.
4. Overlapping objects.

An important feature of the recognition algorithm is its ability to detect the special case of single object mode coupled with good or favourable conditions to truncate the search automatically. We first examine this condition. Under this condition the search can be truncated as soon as X unique features are detected. Under favourable conditions X may be set to a value of 1 or 2, and under good operating conditions to a value of about 4. This strategy would be sufficient if the conditions were known to be good or favourable in advance and it was known that only a single object will be present in the scene. In

this mode, the system operates very fast. However, this mode of operation cannot respond to variations in conditions (i.e. a deterioration of conditions). Therefore the condition for search truncation is altered with practically no extra processing overhead, to allow the system to check automatically whether the special operating condition is met. The truncation algorithm is as follows.

Let R(i) be the number of unique features detected for object i.

Then, if R(i)>T and R(J)=0 for all j≠i then terminate,

else search for all unique features of all objects.

(Note that this condition is tested for after each complete round of unique feature searches i.e. the number of unique features searched for in each object is the same.)

Now the special case of favourable/good operating conditions with a single object in the image can be detected automatically. T must be chosen so that T≧X, and is usually set to be equal to X. The effect of this algorithm is that when T unique features have been detected for object i, if at least one unique feature has been found for another object then the system assumes that either there are multiple objects in the image, or the operating conditions are not good. (i.e. they are poor). Therefore it continues searching for all the features in the unique feature list that remain to be searched, before deciding which objects have been detected. However, it will be noticed that this truncation algorithm is not foolproof. Therefore, a small value for T (i.e. <4) will be used only if the conditions were guaranteed to be favourable, and rapid execution was needed, and faster processing resources could not be provided. However a further safeguard could be used under these conditions by counting the number of unique features

(RN) of object i that were not recognized. Therefore the new termination condition under these special conditions is

Terminate if R(i)=1 or 2, RN(i)=0, and R(j)=0 for all j≠i

Under these conditions the recognition algorithm executes rapidly [50ms on the PDP11/24). Recognition times as low as 10ms have been observed. (See section {5.4} for a discussion of execution time).

The second special case is when the system is provided with good operating conditions, and the objects are presented in one of the first 3 modes. (i.e. without overlapping objects). The termination condition used is

R(i)=4 and R(j)=0 for all j≠i

Under this condition the system operates rapidly if only a single object is present in the image (100-500ms when three objects are being searched for). If more than one object is in the image, all of the features are searched for as before.

In normal operation I use the termination condition with T=6.

i.e. R(i)=6 and R(j)=0 for all j≠i

Finally, when the operating conditions are known to be poor, and possibly with substantial overlapping of objects in the scene, T is set to a large value so that all unique features are always searched for i.e. there is no early termination condition. The system requires about 1s-5s to execute under this condition (or when the termination condition is not met by the other special conditions) when three objects are being searched for. In this situation X remains set to 6, so that any object for which more than 5 unique features are detected is recognized. Objects for which between 3 and 6 features are detected are presented as hypothetical. These hypotheses could be verified by

computing their relational structure (i.e. position and orientation relative to each other), and (or) by searching for other (non-unique) local features of the object. This is not done in the present implementation.

This vision system, then, can be configured to operate rapidly under favourable conditions or flexibly under poor conditions by changing the value of T. Therefore, the system is able to deliver the speed of a binary vision system (with higher reliability) or the flexibility of a grey level vision system by simply choosing the value of T. In practice I would expect the value of T to be set as high as the processing resources allow (in order to achieve the required speed). If it was necessary to have the system run at a constant execution speed, (i.e. by not taking longer to run when the termination condition is not met), the failure of the termination condition could be used to signal rejection. However, such implementation decisions are highly dependent on the specific application.

The flexibility of the vision system arises from the flexibility of the features themselves, and due to the learning algorithm selecting reliable unique features, which means that the system does not have to recognize all of the features for a given object. Thus a large number of features could be lost due to object overlap or degraded operating conditions and still result in a 100% confidence level of recognition.

Because of the reliance on local neighbourhoods, there is essentially little difference between having two objects in the scene that are either overlapping, touching or not touching. In the overlap

situation, small values of object skew and the loss of a large section of the object from view, are the main obstacles to recognition. The section on test data shows that the matching algorithm is relatively insensitive to object skew. This is due to the movement allowed for rep-points within neighbourhoods by the matching algorithm. As far as the loss of unique neighbourhoods due to obscuration is concerned, the program will not be perturbed until less than 6 unique neighbourhoods are left.

A further problem presented by the overlapping and touching situations is that it is possible to create new neighbourhoods from the intersection between features of different objects, which match unique features of objects in the library. However, as shown in section {3.3} the probability of such random matches is low.

Therefore the system is able to recognize overlapping objects provided that

(a) an interesting part of the object remains visible, and

(b) the object plane is not far away from the learned plane.

Once an object is recognized, its position and orientation may be computed. Each detected feature gives a measure of the position and orientation of the object due to the assumption of constant scale. However, in order to obtain an unambiguous measurement of the object position and orientation it is necessary to compute the symmetry of the object during the learning stage. Bolles [1979] discusses a way of computing symmetry. Once the object is registered with the model, it may be inspected by comparing the rep-point descriptions. Perkins [1983] and Barnard [1980] discuss alternative strategies for

inspection.

The recognition algorithm was the easiest to implement due to its simplicity, as all of the complex searching necessary is done by the learning algorithm. The algorithm was implemented so that it would execute rapidly. Therefore, table look-up was used to replace processing whenever possible to speed up computation. A further increase in speed could be achieved by reformatting the feature data as described in section {4.1.3}.

The recognition algorithm is limited to recognizing objects, and does not concern itself with other tasks such as inspection. Objects may be inspected by comparing their rep-point descriptions. In order to do this the rep-point model has to be read into memory once the object is recognized, and the object position and orientation computed. The use of block I/O allows the model data to be read from disc rapidly.

4.5. Comments on the overall system implementation

The programs were all written in Fortran on a PDP11/24 minicomputer. The Extended Instruction Set (EIS) of the PDP11 was used when compiling programs in order to allow rapid execution. (EIS code allowed some programs to run as much as 4 times faster). Assembly code subroutines were not used in any of the algorithms. Therefore, an increase in speed could be achieved by re-coding time consuming parts of the code in PDP11 Macro. This could be especially useful for the recognition algorithm.

The programs were written with a large amount of debug code to allow efficient debugging and improvement of the programs. Some programs contained as much as 60% debug code. This scheme allowed me to write the programs so that rapid execution and good diagnostics were available. In normal execution the programs are compiled without the debug code. In this mode only the essential processing for the algorithms to execute properly is performed. This allows rapid execution of the programs. In program development mode the debug code is compiled as well. This code which is interleaved with the algorithm code generates a dynamic display of algorithm execution on the user VDU, and computes a variety of statistics. The debug displays are controlled by a debug status vector which allows different parts of the display to be enabled and disabled, so that the user may suppress displays that are not of interest at the time. When all of the display is enabled, the program spends most of its time redrawing the screen. Therefore a program -such as the rep-point program- that executes normally in 14s takes many hours to execute. In order to allow the programmer to find the parts of the execution sequence of interest, a degree of status pattern matching is also incorporated. The program then continues execution without displaying any information until the specified program status is found. This scheme allowed efficient debugging of programs. Further details of the user interface to the software is given in Appendix 2.

It should be stated, however, that the main source of information for program debugging comes from displaying the processed images on a display monitor. Although it is not possible to prove the correctness of programs by this method, it allows incorrect programs to be

detected very quickly. This gives us a tremendous advantage over other areas of computing, where assessing program correctness can be a major problem. Thus, a good quality display device is invaluable during program development.

The present implementation can be improved in several ways to achieve better results, and faster processing. The implementation of the rep-point algorithm has a flaw due to an asymmetry in the way the horizontal and vertical directions are handled. This should be removed. An alternative implementation using relaxation techniques may also be possible. The rest of the pre-processing algorithms do not need any special improvements, but of course they should be implemented in special purpose hardware. The recognition algorithm should be rewritten so that feature property values do not have to be unpacked.

These are the improvements needed within the unextended architecture. However, future implementations should attempt to code the proposed architectural extensions {section 6.1} as well in order to allow the limitations of the presently implemented architecture to be removed.

Chapter 5

Tests and Results

The purpose of this chapter is to demonstrate that the strategy developed in chapters 2 and 3 is usable, and to establish the performance of the system in terms of the objectives set out in chapters 1 and 3. In particular, this chapter aims to indicate the extent of the flexibility achieved by the system.

This chapter is organized as follows: Section {5.1} introduces the system test strategy. Section {5.2} describes the overall system tests, which is followed by a discussion of tests performed on the individual parts of the system {section 5.3}. This is followed {section 5.4} by a discussion of system execution speed. Section {5.5} gives a summary of the test results.

5.1. Introduction

Testing vision systems is a difficult task. This is due to the badly understood nature of the problem, and the huge number of possibilities that would have to be considered if a vision system were to be tested exhaustively; in fact, the number of possible input patterns may be considered infinite for all practical purposes (e.g. a 256x256 8-bit image can take $256^{256x256}$ distinct patterns!) There are two ways of tackling this problem:

1. Test the system to determine that the design objectives have been met by imaging a few objects and verifying recognition.
2. Use the knowledge of the system design strategy to map the different failure modes of the system by selecting tests that are most likely to result in failure.

All systems that the author is aware of use the first strategy; most published papers present a few image instances that the system successfully recognized, so that the judgement of the merits of the algorithms must depend almost wholly on the knowledge of the design strategy. In this thesis, the second test strategy is used. (Note that this includes the first). The system is tested by mapping its performance when its basic assumptions are not met. That is, each of the assumptions made by the system (e.g. constant lighting) is gradually (and independently) varied until recognition fails. The result is a map of the system sensitivity to each of its assumptions. In order that the different tests be comparable, a single set of objects was used.

It should be noted that all thresholds were kept constant throughout these tests. (That is, thresholds were not changed in order to get the best performance for each test). Further, the optical parameters were also kept constant i.e., the lens aperture was not changed to obtain the best exposure for each object. The aperture was chosen to allow reasonable imaging of all the objects. This means that some objects were imaged less well due to variations in overall object reflectance. These tests verify an important characteristic of the feature descriptor: It will be recalled that the extended learning algorithm confirms the reliability of unique features only over variations of 2D position and orientation of the object. Therefore, the system does not know whether these features are structural through other variations such as scale. Thus these tests establish the extent to which the pre-processor and the feature selection process are able to select features that are structural outside the sampled range of images i.e. if the feature descriptor did not show a degree of uniformity we would expect that when object A was in the image, the system would detect unique features (UFs) of objects B and C as well. Such features will be referred to as spurious features. The ideal performance would be for the number of UFs found for object A to decrease (i.e. graceful degradation) when the operating conditions degrade, while the number of UF found for B and C remain at 0. It should be noted that these tests are far more stringent than those reported on most previously published systems, as the system is being tested outside its design limits to failure.

5.1.1. Selection of Test Objects

The tests performed on the system can be divided into two categories with respect to system sensitivity to the actual objects used:

1. Tests with the object set in Fig. 5-1, and
2. Tests with other objects.

The second class of tests may be considered another test of varying one of the parameters of the system i.e. the 'parameter' of the actual objects used. Thus, the tests based on all other parameter variations were done using the object set in Fig. 5-1. The only criterion used when selecting this test set was that the objects should have some 'local structural activity'. That is, since the implemented system was based on the use of local features, it was clearly necessary that such

Fig. 5-1 The three test objects

features be available on the objects. It would have been of little value to use objects with global unique features only (e.g. a rectangular shape), as the present architecture explicitly excludes such objects. (See chapter 6 for a discussion of how this limitation may be removed).

Historically, the three objects in Fig. 5-1 were the first that I found. The apparent similarity between the objects (to us), only serves to make recognition harder, as object recognition depends on object dissimilarity rather than similarity. However, any doubt about whether the system is able to recognize only these three objects should be dispelled by the additional tests which were performed with other objects (see sections {5.2.3.7, 5.2.3.8, 5.2.3.9}).

5.1.2. Test Procedure The three objects in Fig. 5-1 were taught to the system. These three objects will be referred to as cutter, gear wheel, and tooth wheel (from left to right). This object library will be referred to as the CTG library. The test environment is shown in Fig. 5-2. A single object was then introduced and one of the parameters was varied until the threshold of recognition was reached (i.e. only 6 UFs were detected for the object in the scene). All of the other conditions were kept constant. The objects were presented on a black background for maximum contrast. The lighting was kept diffuse using the set up shown in Fig. 5-2. Recognition was defined to have failed when either

(a) less than 6 UF of object A was detected when A was in the image, or

(b) more than 6 UF of object B was detected when B was not in the

Fig. 5-2 The environment for the set of controlled tests.

image.

5.1.2.1. Forming the test library

The three objects were taught to the system by showing each object in 5 randomly chosen 2D positions and orientations in the image. This resulted in 15 images named CUTTER1-5, TOOTHW1-5 and GEARW1-5. Then, each image was processed as follows: Firstly, the edge image was formed, followed by a rep-point image. Fig. 5-3 is an exam-

ple of an edge image which was formed from image CUTTER1. Fig. 5-4 is the rep-point image formed from Fig. 5-3.

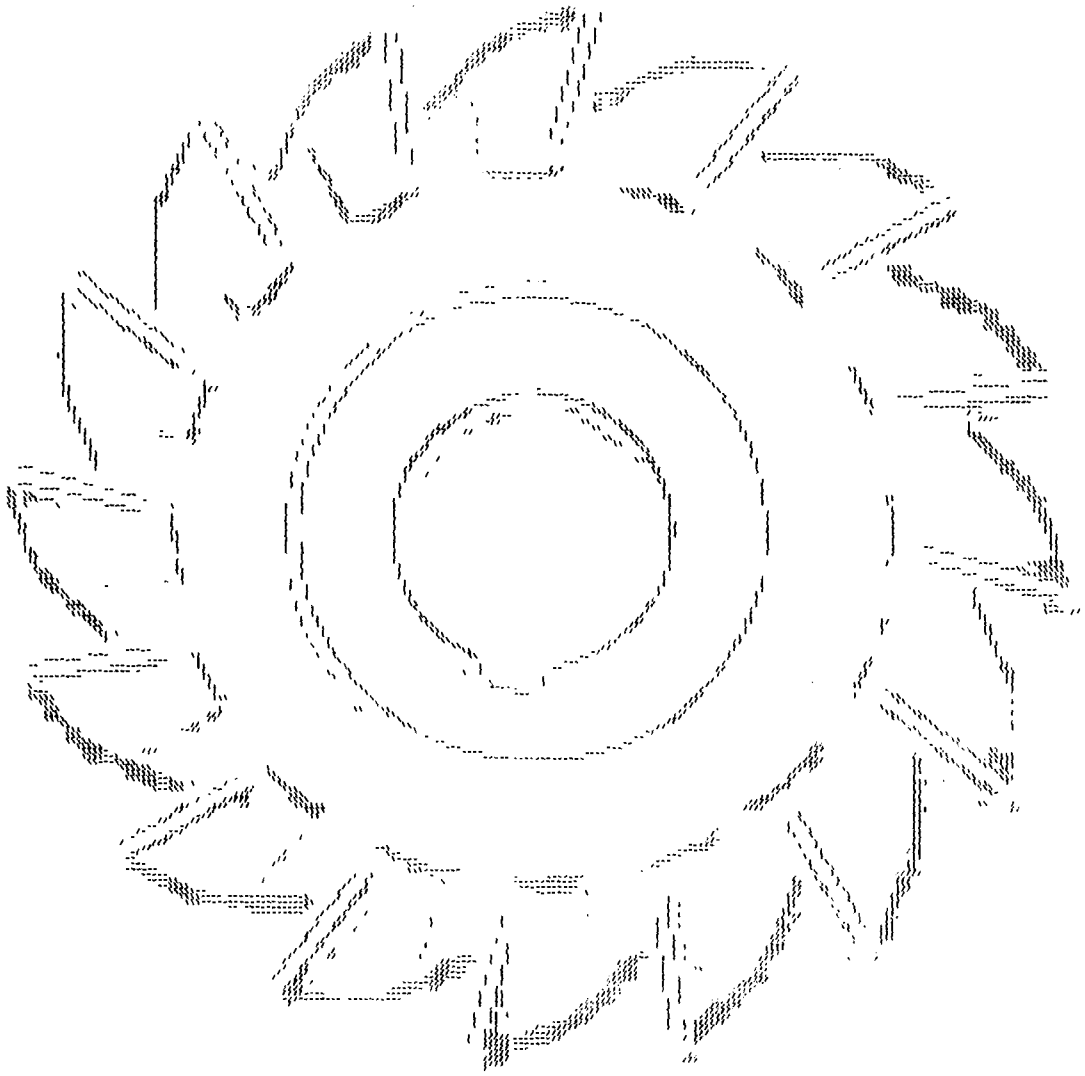


Fig. 5-3 Edge image of the cutter in Fig. 5-1 (See also Fig. 5-7)

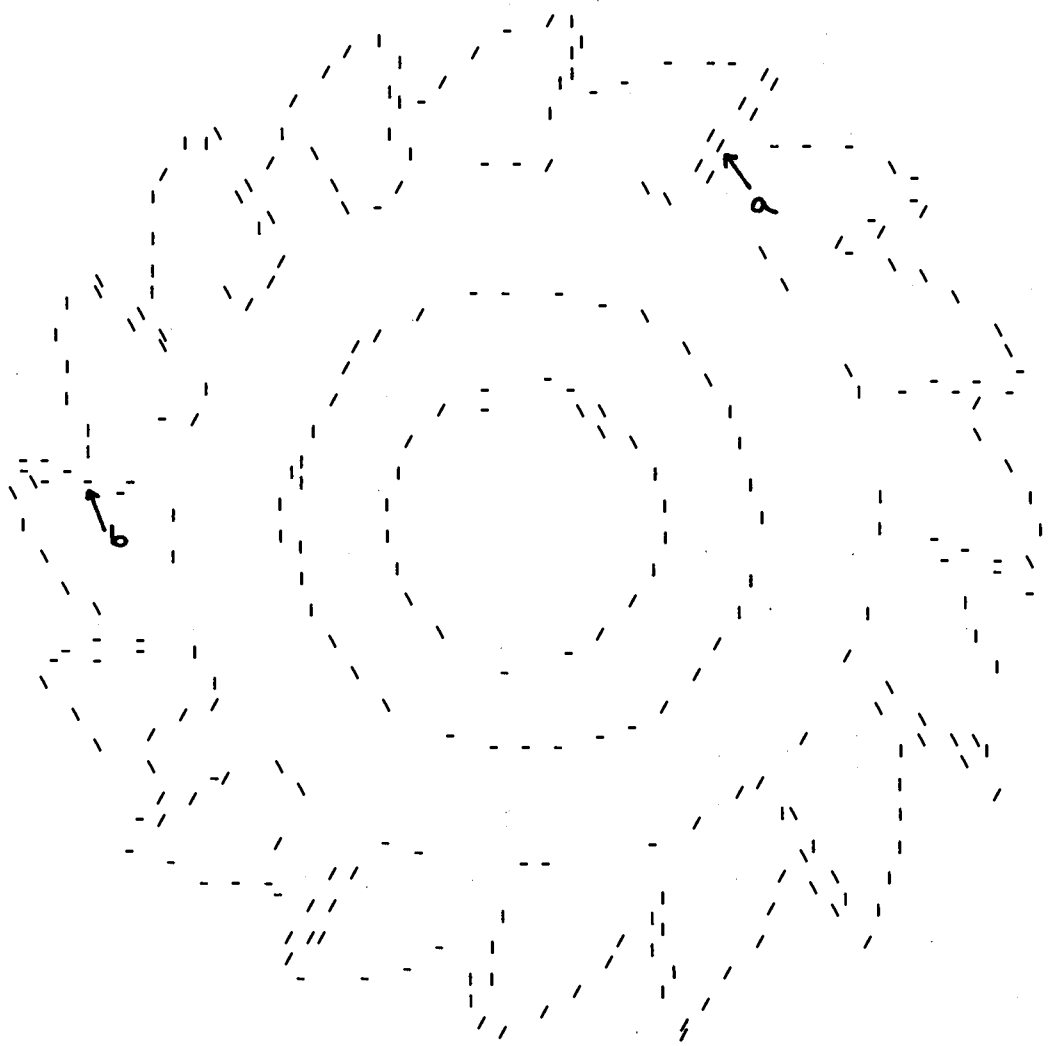


Fig. 5-4 Rep-points found in Fig. 5-3

The rep-point image was used to construct a relational model of the object based on a relational structure of rep-points, and a relational structure of local features. (The relational structure of rep-points is not used by the system at present).

Next, the learning algorithm was run on these images to obtain unique features for each object. The following table shows the progress of the learning algorithm.

(The first and second columns show the names of the models being matched. The third column shows the number of local features from the model in the first column that were searched for in the second model. The fourth column shows the number of features that were found.)

```
Find reliable features for CUTTER
CUTTER1          CUTTER2          381      225
CUTTER1          CUTTER3          225      182
CUTTER1          CUTTER4          182      145
CUTTER1          CUTTER5          145      130
Number of reliable features for CUTTER=130

Find UF for CUTTER. Compare with TOOTHW and GEARW
CUTTER1          TOOTHW1          130       50
CUTTER1          TOOTHW2           80        6
CUTTER1          TOOTHW3           74        2
CUTTER1          TOOTHW4           72        3
CUTTER1          TOOTHW5           69        1
68 features unique to CUTTER compared with TOOTHW
CUTTER1          GEARW1           130       41
CUTTER1          GEARW2            89       23
CUTTER1          GEARW3            66        9
CUTTER1          GEARW4            57        1
CUTTER1          GEARW5            56        1
55 features unique to CUTTER compared with GEARW
51 features are unique to CUTTER compared with TOOTHW and GEARW.

Find reliable features for TOOTHW
TOOTHW1          TOOTHW2          317       80
TOOTHW1          TOOTHW3           80       56
TOOTHW1          TOOTHW4           56       53
TOOTHW1          TOOTHW5           53       48
Number of reliable features for TOOTHW=48
```

```
Find UF for TOOTHW. Compare with CUTTER and GEARW
TOOTHW1          CUTTER1          48       20
TOOTHW1          CUTTER2          28        0
TOOTHW1          CUTTER3          28        0
TOOTHW1          CUTTER4          28        0
TOOTHW1          CUTTER5          28        0
28 features unique to TOOTHW compared with CUTTER
TOOTHW1          GEARW1           48       16
TOOTHW1          GEARW2           32        0
TOOTHW1          GEARW3           28        0
TOOTHW1          GEARW4           28        0
TOOTHW1          GEARW5           28        0
28 features unique to TOOTHW compared with GEARW
27 features are unique to TOOTHW compared with CUTTER and GEARW.


Find reliable features for GEARW
GEARW1           GEARW2          384       91
GEARW1           GEARW3           91       47
GEARW1           GEARW4           47       23
GEARW1           GEARW5           23       18
Number of reliable features for GEARW=18

Find UF for GEARW. Compare with CUTTER and TOOTHW
GEARW1           CUTTER1          18        8
GEARW1           CUTTER2          10        3
GEARW1           CUTTER3           7        0
GEARW1           CUTTER4           7        1
GEARW1           CUTTER5           6        0
6 features unique to GEARW compared with CUTTER
GEARW1           TOOTHW1          18        7
GEARW1           TOOTHW2          11        0
GEARW1           TOOTHW3          11        0
GEARW1           TOOTHW4          11        0
GEARW1           TOOTHW5          11        0
11 features unique to GEARW compared with TOOTHW
 6 features are unique to GEARW compared with CUTTER and TOOTHW.


The final result was
        Reliable        Unique                  Unique between two
CUTTER     130           51                      -       68       55
TOOTHW      48           27                     28        -       28
GEARW       18            6                      6       11        -
```

The matrix on the right is the number of UF for each object when compared with one other object. This matrix demonstrates the difference between any two objects. It may be used to cluster objects {section 6.1.4} and to improve the speed of the learning algorithm during

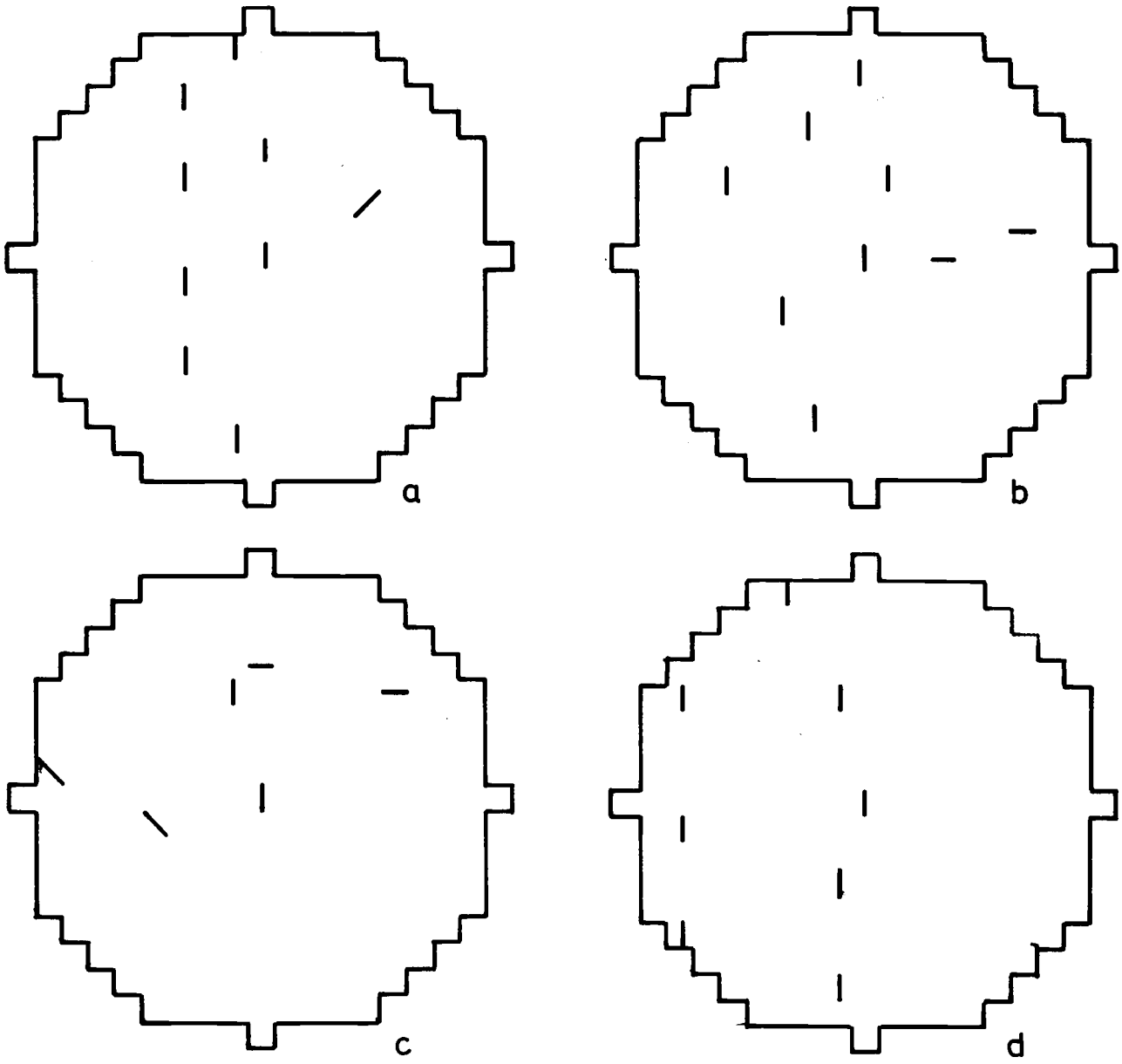


Fig. 5-5 Four of the unique features chosen for the CTG library. (a) and (b) are from the cutter, (c) is from the tooth wheel (d) is from the gear wheel.

incremental learning. The UF sets are the intersection of the sets in this matrix. Fig. 5-5 are examples of four of the unique features chosen by the system. The positions of UF(a) and UF(b) are marked on Fig. 5-4 with arrows. UF(c) was taken from the outer edge of the tooth wheel and shows one of the teeth. UF(d) shows one of the teeth

of the gear wheel.

Due to the fixed size of the data structures used, only 31 UF were used for the cutter during recognition. All UF were used for the other two objects (i.e. 27 and 6 respectively). The above table demonstrates the way the learning algorithm operates. The reader will notice the way the number of reliable features that match features in the other objects drop rapidly after the first instance is matched, so that only a small number of object instances are required for learning.

In the following discussion, a naming convention has been adopted for object models to indicate the object name and the kind of test being performed. There are 3 fields as follows:
<Object name> <Type of test> <Instance number>
Object names and test type are often abbreviated for convenience e.g. CUTL1 is an image of the cutter for light variation tests, and TOOTHZ3 is the tooth wheel with the zoom changed (i.e. scale change).

During the system tests, the recognition stage was allowed to search for all unique features. Thus, there was no early termination condition {section 4.4}. Therefore, the recognition statistics given should be interpreted as follows: Any object for which 6 or more UF are detected is considered recognized by the system, while the detection of 6 or more spurious features for a single object constitutes a mismatch.

5.2. Overall System Tests

Four types of tests were performed on the overall system.

1 Tests to verify the sensitivity of the system to the three basic assumptions.

2. Tests to verify sensitivity to implicit assumptions.

3. Tests over other imaging conditions.

4. Tests on the system response to variation of internal operating parameters.

5.2.1. Sensitivity to the Three Basic Assumptions

The system was first tested by allowing the three basic constancies expected by the system to vary i.e. through variations of lighting, scale, and object plane. This was done by changing one of the parameters until the system failed to recognize the object in the scene.

5.2.1.1. Light Variation Test In this first test, the light intensity was reduced until the system failed to recognize the object in the image (i.e. less than 6 UF were detected). I found it difficult to control the lighting accurately. The effective intensity was varied by stopping down the aperture of the lens. The light intensity was measured by summing the pixel data over a central square of the image. It will be noticed that reducing the aperture also increases the depth of field. However, this has a minimal effect as the object is always in focus. (Distance from the camera to object was about 1.2m, while the object height was about 1cm-5cms. Initial lens aperture was f2.8).

Fig. 5-6 shows the variation in the number of UFs found when the cutter was in the image. Note that the number of UFs shown include multiple detections of some features due to object symmetry. The following is the list of UF found for each instance of the cutter as the light intensity was reduced. Note the spurious features detected for the gear wheel. This turned out to be a high level of spurious features for the system. It will be noticed that the number of spurious features remained less than 6, so that the highest (spurious) confidence level reached was 75%. The list on the right shows the number of UF found when the tooth wheel was in the image. Note the very low

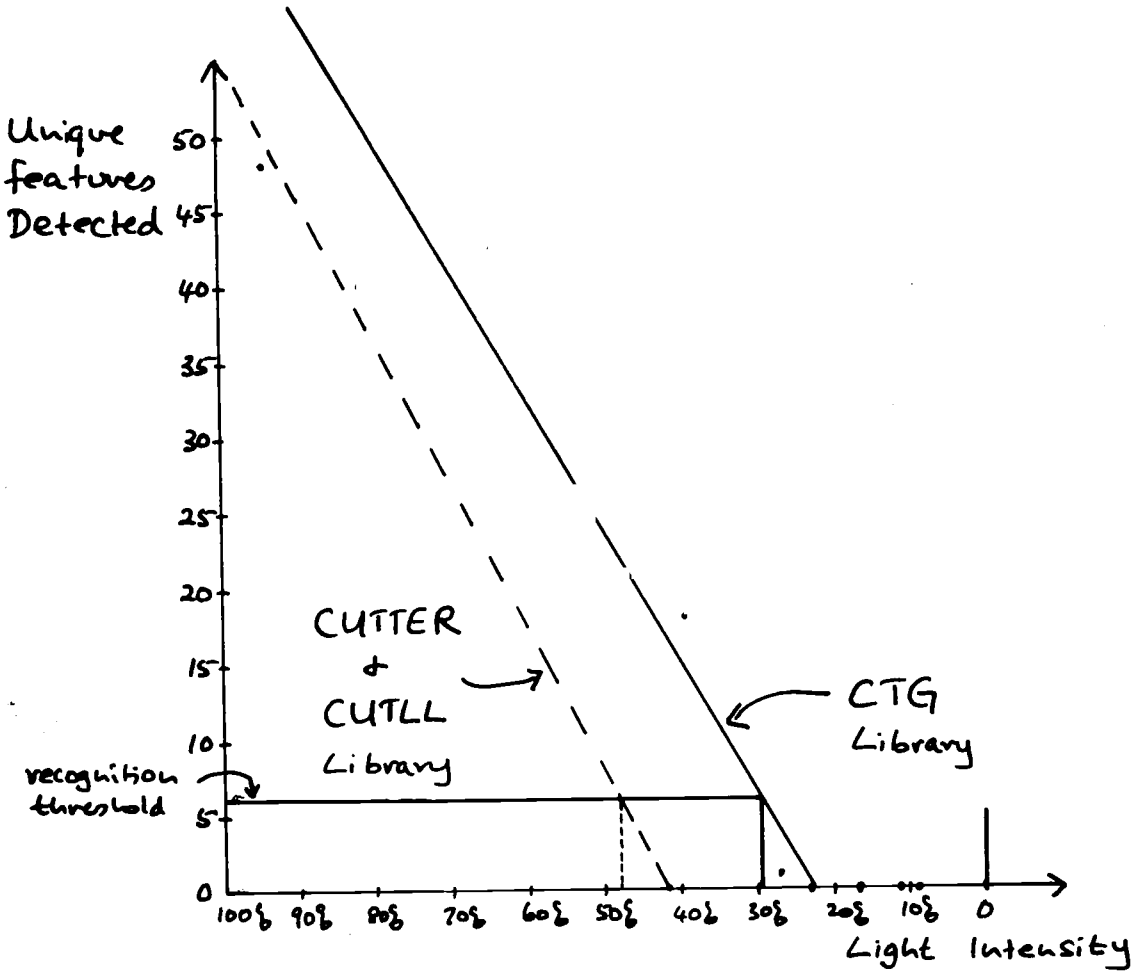


Fig. 5-6 Unique features detected for cutter with light intensity

level of spurious features.

(Column (1)- instance number, Column(2)- light intensity summed over a central square, Column(3)- UF found for cutter, Column(4)- UF found for tooth wheel, Column(5)- UF found for gear wheel).

Intensities of original images were:
CUTTER - 4849
TOOTHW - 7349

CUTL 1-17

(1)	(2)	(3) C	(4) T	(5) G
1	4563	48	0	0
2	4182	57	0	4
3	3950	40	0	2
4	3679	58	0	5
5	3401	47	0	3
6	3088	47	0	1
7	2856	42	0	2
8	2660	25	0	4
9	2381	20	0	1
10	2182	22	0	2
11	1916	18	0	1
12	1630	3	0	1
13	1344	1	0	3
14	1108	0	0	1
15	809	0	0	0
16	534	0	0	0
17	458	0	0	0
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				

TOOTHL 1-31

(2)	(3) C	(4) T	(5) G
7349	0	89	0
6961	0	78	0
6853	0	47	0
6490	0	98	0
6354	0	83	0
6046	0	86	0
5823	0	84	0
5439	0	91	0
5281	0	109	0
5003	0	114	0
4712	0	121	0
4485	0	120	0
4279	0	113	0
3986	0	109	0
3739	1	135	0
3435	0	110	0
3212	0	127	0
2926	0	132	0
2635	0	86	0
2426	0	38	0
2196	0	44	0
2009	0	58	0
1750	0	23	0
1606	0	21	0
1494	0	9	0
1316	0	0	0
1069	0	0	0
891	0	0	0
638	0	0	0
517	0	0	0
369	0	0	0

It will be noticed from the graph that the recognition failed when the intensity was 70% below the intensity at which the cutter was learned. (The corresponding figure for the tooth wheel was 81%).

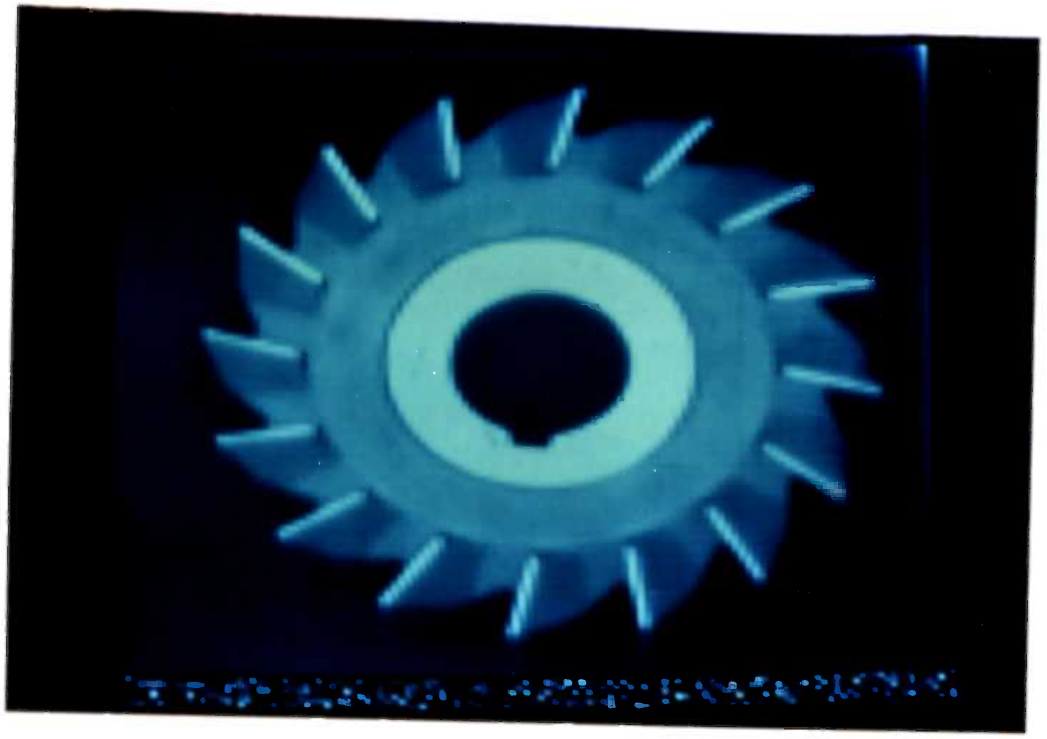


Fig. 5-7 The cutter as it was learned by the system
Note: Image resolution is 128x128 (for display only)

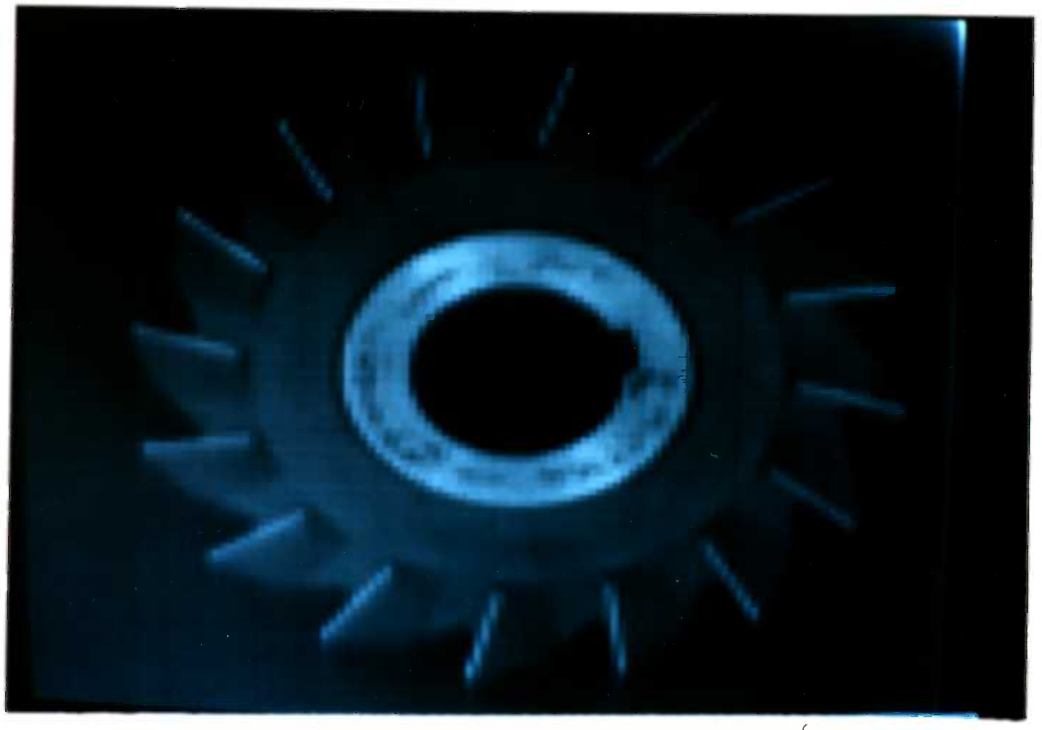


Fig. 5-8 The cutter at the threshold of recognition (light variation)

Fig. 5-7 shows one of the images of the cutter as it was learned, and Fig. 5-8 shows the contrast level at which the system just recognized the object. In order to check the importance of the edge threshold in this particular test the edge threshold was varied proportional to the measured brightness of the image. As expected, this improves the recognition threshold, and allows the cutter to be recognized with the light intensity 78% down (86% for tooth wheel), while the number of spurious features remain low despite the increase in the amount of noise that gets past the edge detector. This increase in noise content can be seen in Fig. 5-9 which shows the rep-points that were generated at the lower edge threshold when the cutter was just recognized.

(1- instance number, 2- intensity, 3- edge threshold used (normal threshold is d=80, k=0.75. k was not changed), 4- UF detected.)

CUTL 14-17

(1)	(2)	(3)	(4)	
				G
14	1108	18		2
15	809	14	3	3
16	534	9	1	0
17	458	8	0	1

TOOTHL 25-31

			C	T	G
25	1494	17	0	7	0
26	1316	15	0	13	0
27	1069	12	0	0	0
28	891	10	0	12	0
29	638	7	0	1	0
30	517	6	0	4	0
31	369	4	0	0	0

In order to demonstrate the possibility of artificially adjusting the

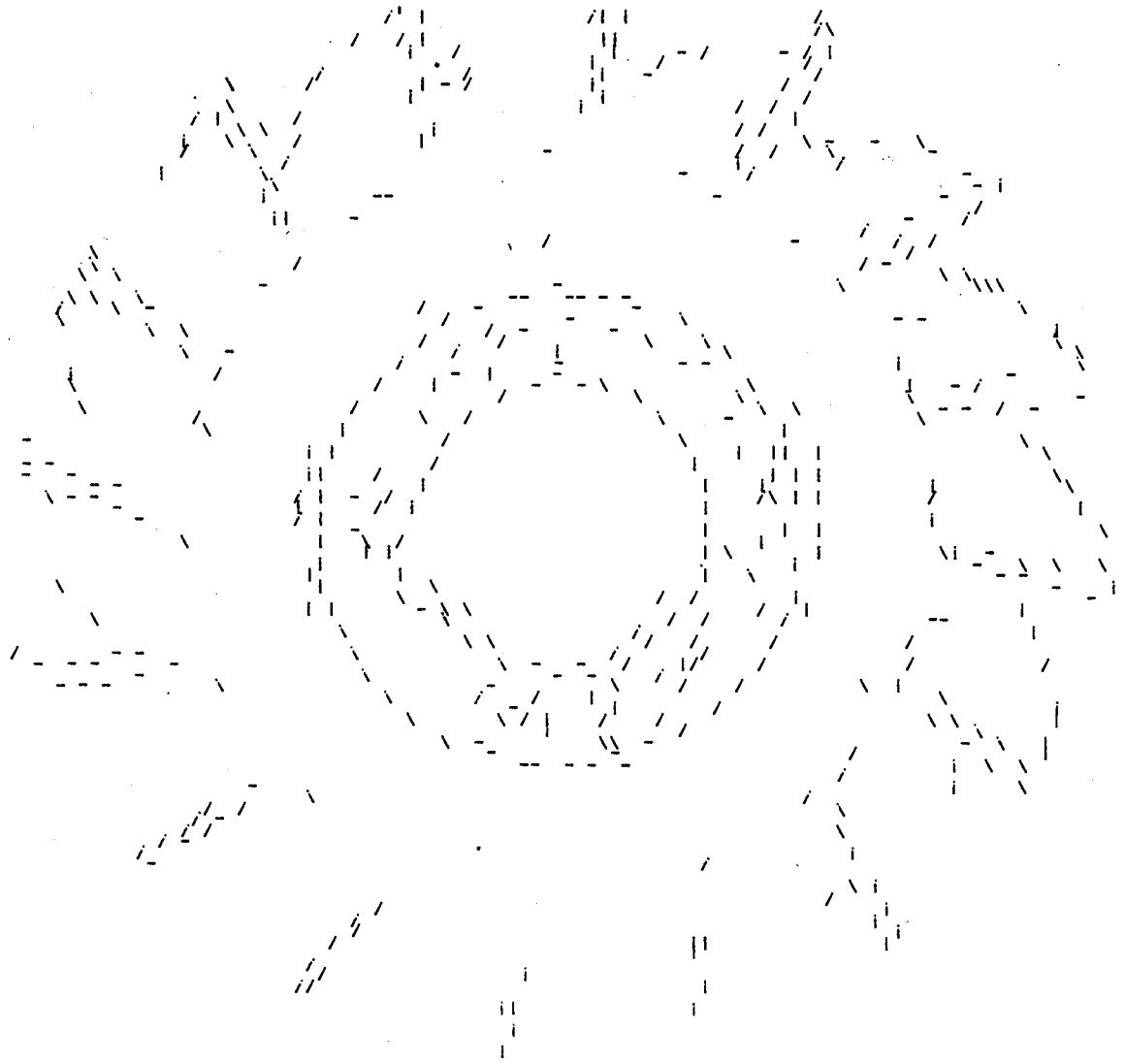


Fig. 5-9 The cutter at the threshold of recognition (rep-point image) Edge detector thresholds were lowered proportional to average intensity d=18, k=.75

system flexibility, the system was taught the cutter at normal intensity, and the cutter at a lower intensity (44% of original) as separate objects. The learning statistics were as follows.

	Reliable	Unique
CUTTER	130	32
CUTLL	58	1

This library was used to re-recognize the data for the cutter with light variation. This reduced the recognition flexibility to 48% of original intensity. (See the broken line in Fig. 5-6. Note that the points for this graph have not been plotted). The statistics were:

(1- instance number, 2- intensity, 3- UF for CUTTER, 4- UF for cutter at low light intensity -CUTLL).

CUTL 1-17

(1)	(2)	(3) C	(4) CLL
1	4563	49	2
2	4182	41	0
3	3950	35	2
4	3679	37	2
5	3401	24	3
6	3088	26	4
7	2856	13	6
8	2660	5	4
9	2381	7	2
10	2182	0	8
11	1916	0	6
12	1630	0	2
13	1344	2	1
14	1108	2	0
15	809	1	0
16	534	0	0
17	458	0	1

The original CTG library was used to recognize the CUTLL set of images (which are at approximately 44% of the original lighting level).

CUTLL 1-5

	C	T	G
1.	6	0	1
2.	12	0	1
3.	8	0	1
4.	15	0	3
5.	14	0	1

(Note: There was an inadvertent fluctuation in scale on this set of images)

5.2.1.2. Scale Variation Test

In this test the size of the object was changed by varying the zoom of the lens. All other parameters were kept constant. The lighting too was kept constant as far as possible, although this was not easy. Fig. 5-10 shows the variation of UFs that were detected. The threshold of recognition was reached when the object was 39% smaller than in the learned image (43% for the tooth wheel). Fig. 5-11 is the image of the cutter at the threshold of recognition. The recognition statistics were as follows: (Note the low level of spurious features despite operation well outside the original design limits.)

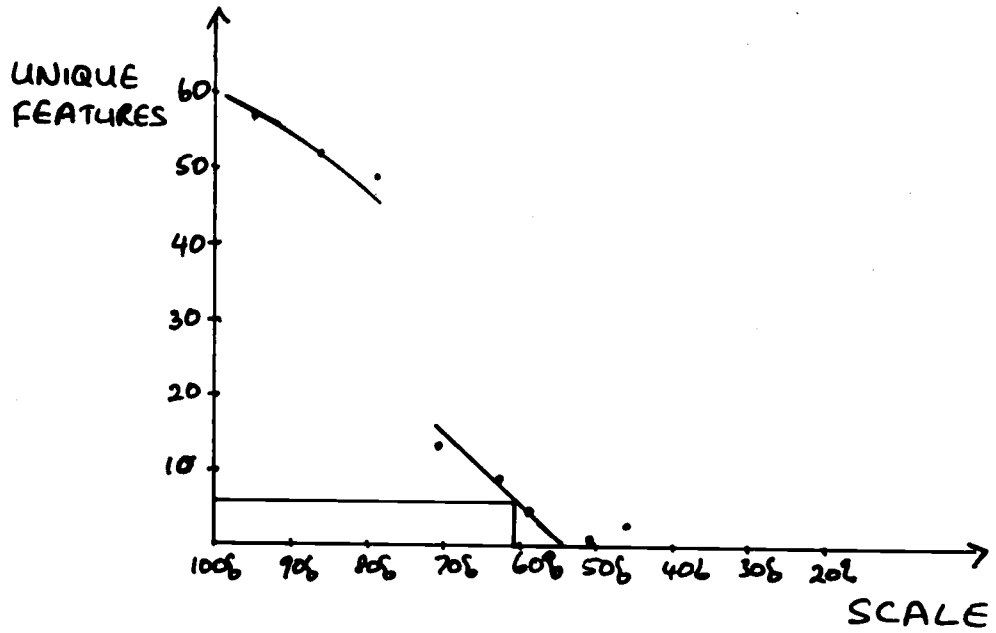


Fig. 5-10 Unique features detected for the cutter with scale variation

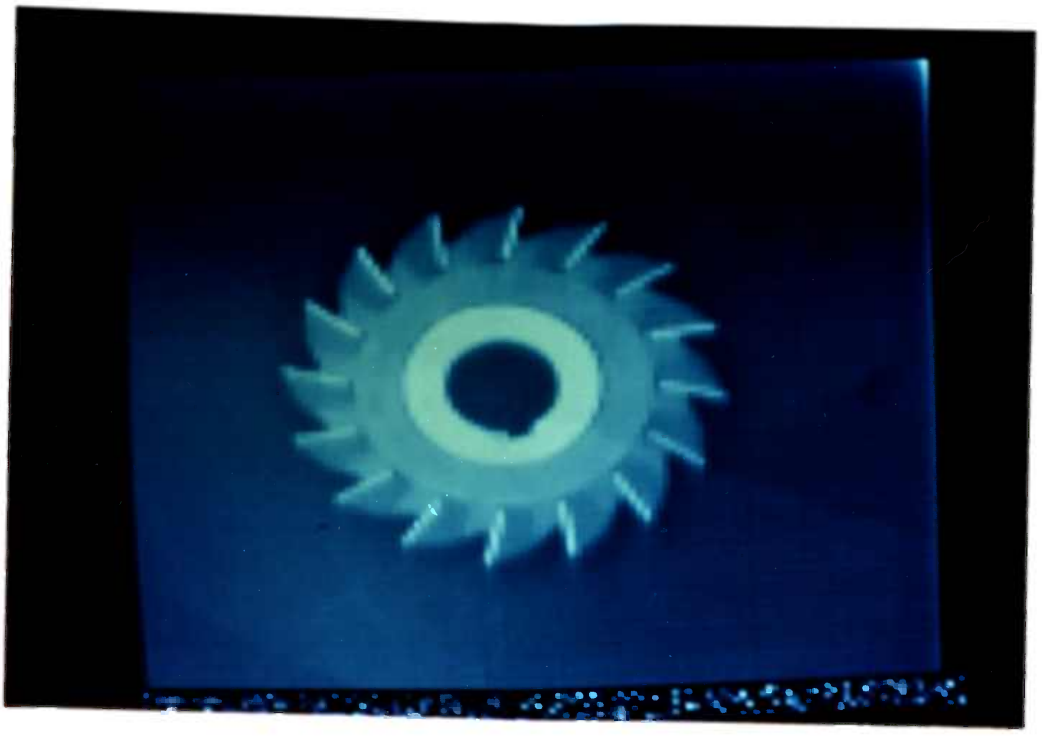


Fig. 5-11 The cutter at the threshold of recognition (scale reduction)

Size of original images were:
CUTTER - 6.3
TOOTHW - 6.2

CUTZ 1-8

Size	C	T	G
6.0	57	0	0
5.45	52	0	0
5.0	49	2	0
4.5	13	2	0
4.0	9	0	0
3.7	5	0	0
3.25	1	0	3
2.9	3	0	5

TOOTHZ 1-6

Size	C	T	G
5.5	0	67	0
5.1	0	47	0
4.6	0	16	0
4.0	0	14	0
3.45	0	2	0
3.0	0	1	0

Once again, in order to check the possibility of changing the system flexibility artificially, the system was taught the original cutter images and 5 other images of the cutter at a smaller scale as separate objects (57% of normal size). The learning statistics were as follows:

	Reliable	Unique
CUTTER	130	44
CUTSM	21	0

The CUTZ images were then re-recognized using this library to check the change in system flexibility.

CUTZ 1-8

Size	C	CSM
6.0	50	0
5.45	46	0
5.0	42	0
4.5	7	0
4.0	0	0
3.7	1	0
3.25	0	0
2.9	0	0

This demonstrates that the recognition flexibility was reduced to about 30%. Recognition on CUTSM itself using the original CTG library resulted in the following figures: (This is a scale reduction of 43%. Note the low level of spurious features).

CUTSM 1-7

	C	T	G
1	7	0	1
2	2	0	0
3	4	0	1
4	1	0	0
5	6	0	0
6	3	0	1
7	2	0	0

5.2.1.3. 3D Orientation Variation In this test the object was tilted out of the learned plane by placing it on an inclined metal sheet. Fig. 5-12 shows the variation in the number of UFs detected. The recognition threshold was reached when the cutter was 32° off the

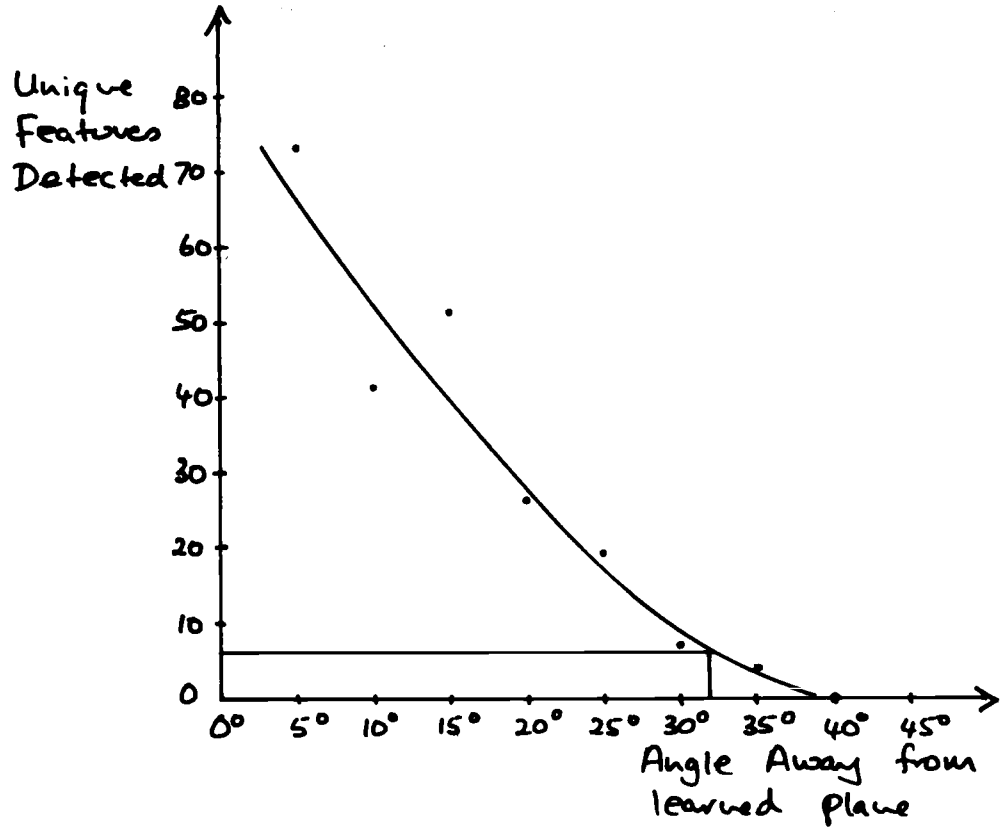


Fig. 5-12 Unique features detected for the cutter with orientation variation

learned plane (47° for the tooth wheel). Fig. 5-13 shows the cutter on the inclined plane at the threshold of recognition. From the following statistics it will be noticed that the number of spurious features were very high on this test. (But note the lack of spurious features when the tooth wheel was in the image). This demonstrates the need to extend the reliability check to these situations if the system is required to operate with such images.

	CUTI 1-8		
Angle	C	T	G
5°	73	1	2
10°	41	1	0
15°	51	1	0
20°	26	17	1
25°	19	7	0
30°	7	14	2
35°	4	6	1
40°	0	4	2

	TOOTHI 1-11		
Angle	C	T	G
5°	0	108	0
10°	0	87	0
15°	0	75	0
20°	0	54	0
25°	1	101	0
30°	0	73	0
35°	0	59	0
40°	0	20	1
45°	0	7	0
50°	0	4	0
55°	0	3	0

5.2.2. Variation of other Implicit Assumptions

As pointed out in section {3.2.1} there are a large number of implicit assumptions such as proper camera operation which the system cannot be tested for. However, two tests were carried out in this category.

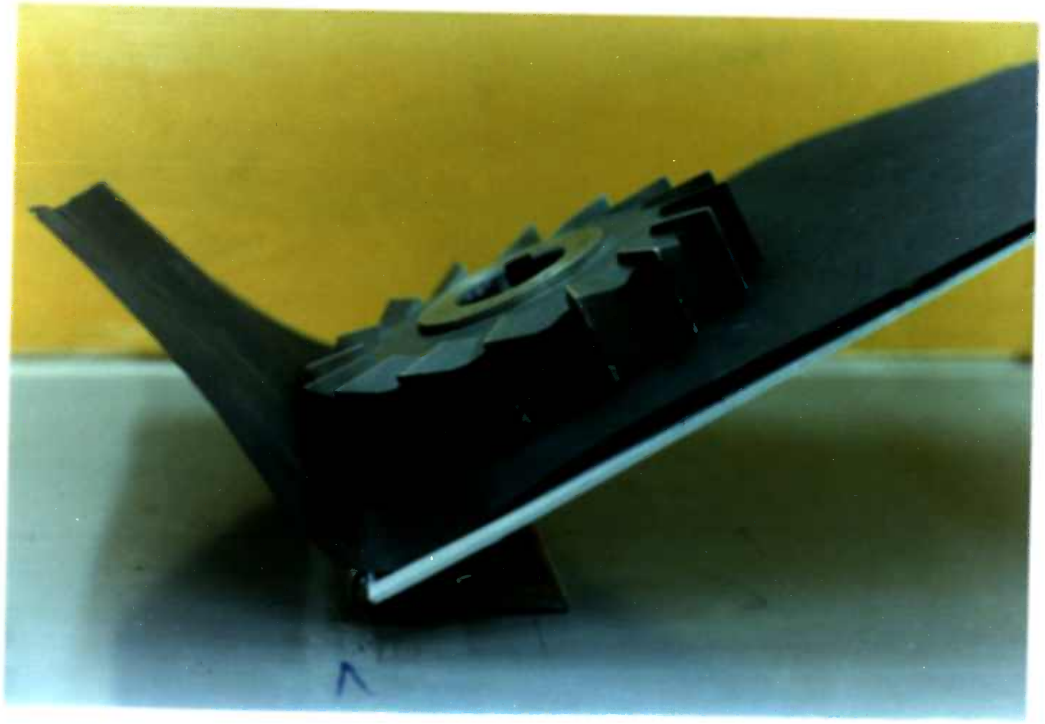


Fig. 5-13 The cutter at the threshold of recognition {3D orientation var.}

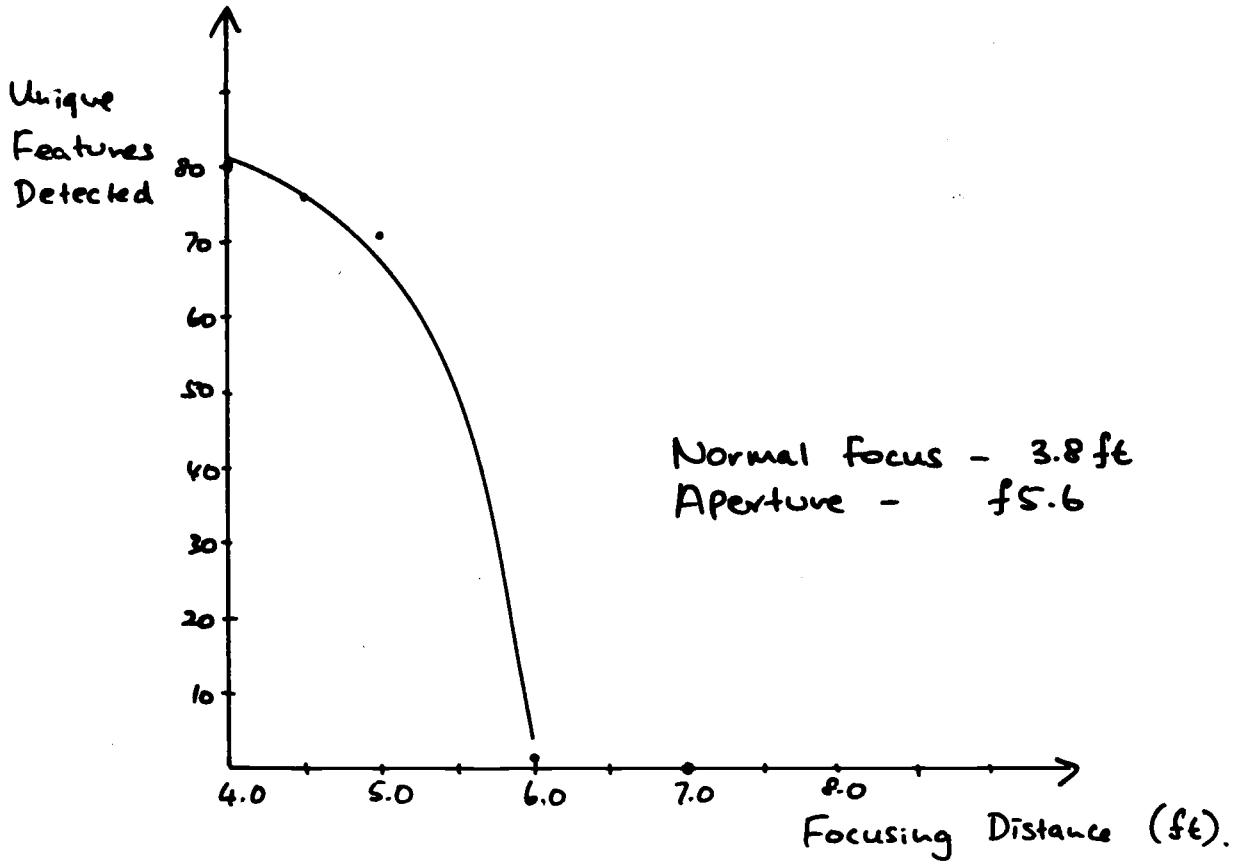


Fig. 5-14 Unique features detected for the tooth-wheel with defocussing

5.2.2.1. Camera Defocussing The system was able to cope with a small amount of image defocussing. Fig. 5-14 shows the variation in the number of UFs detected for the tooth wheel when the lens focus was changed from the correct value of 3.8ft. The aperture was f5.6. These images were taken in ordinary lighting conditions {section 5.2.3.1}. Recognition statistics were:

TOOTHBLUR 1-5

Focusing distance	C	T	G
4 ft	0	80	0
4.5 ft	0	76	0
5.0 ft	0	71	0
6.0 ft	0	1	0
7.0 ft	0	0	0

The focusing distance was read off the lens.

5.2.2.2. Gaussian Noise Gaussian noise was added to the input images artificially until the system failed to recognize the object in the image. The noise was generated using an algorithm given by Knuth [1969] p.104. (Note that the sum was truncated to 8 bits). The signal to noise ratio was measured using the definition given by Pratt [1978] p.498.

i.e., signal to noise ratio $=\frac{h^2}{\sigma^2}$

where h is the edge height and σ is the standard deviation of the noise. The edge height was measured by taking the average edge strength of the edges that are detected when no noise was added. Therefore the signal to noise ratio is the signal to noise ratio seen by the vision system at the chosen edge threshold. Fig. 5-15 shows the variation in the number of UFs that were detected when the signal to noise ratio was decreased. The recognition threshold was reached when the signal to noise ratio was 7 (8.5 dB). The corresponding value for the tooth wheel was 4 (6 dB). The statistics were as follows: (Once again note the lack of spurious features despite the random noise that was added. This supports the calculation for random match probability in appendix 1).

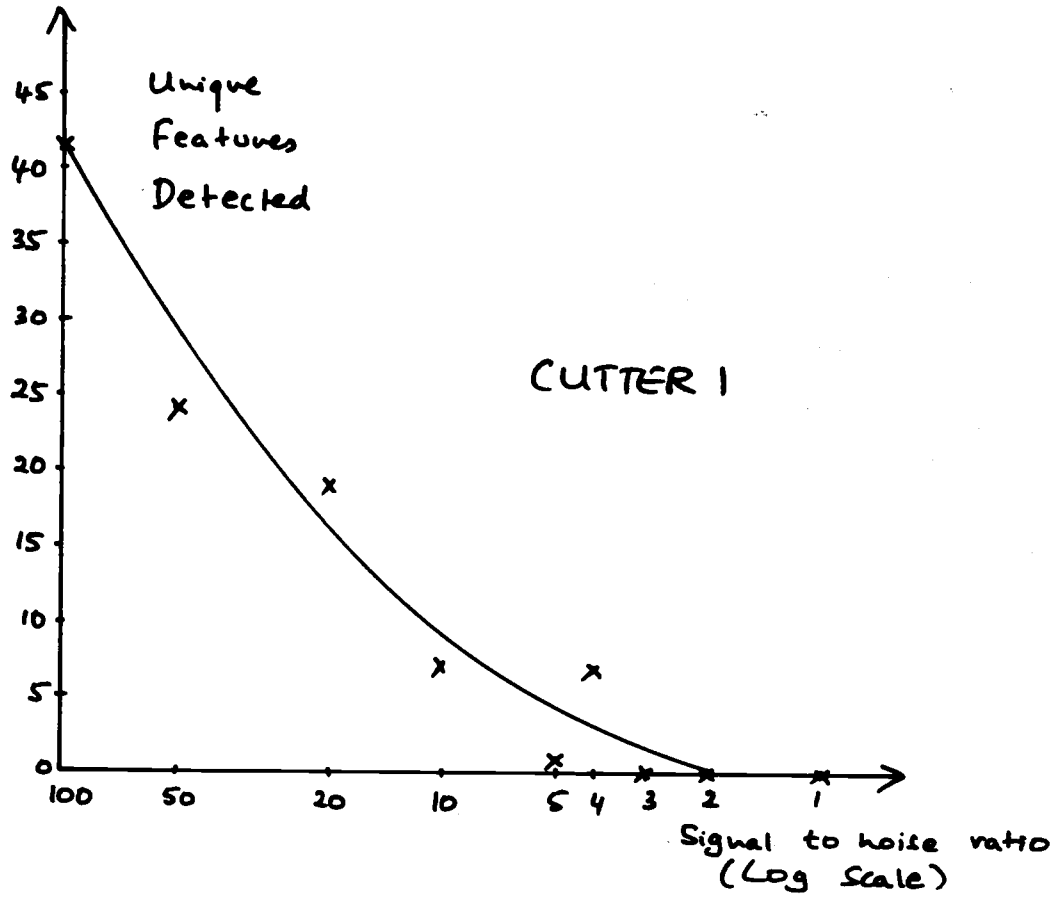


Fig. 5-15 Unique features detected for the cutter with Gaussian noise

CUTTER1
Average edge strength of edges that are normally detected for CUTTER1=33.25

σ	S/N ratio	C	T	G
3.32	100	43	0	0
4.71	50	24	2	1
7.44	20	19	0	0
10.52	10	7	0	1
14.88	5	1	0	0
16.64	4	7	0	0
19.21	3	0	0	1
23.53	2	0	0	0
33.28	1	0	0	0

TOOTHW1
Average edge strength of edges that are normally detected for TOOTHW1=40.25

σ	S/N ratio	C	T	G
4.03	100	0	44	0
5.69	50	0	27	0
9.00	20	0	32	0
12.73	10	0	30	0
18.00	5	0	9	0
20.13	4	0	7	0
23.24	3	0	3	0
28.46	2	0	0	0
40.25	1	0	3	0

Fig. 5-16(a) shows the rep-points that were found for the cutter when the signal to noise ratio was 4. In Fig. 5-16(b), the normal requirement that each rep-point be nominated by at least 2 edge points has not been enforced. This shows the dramatic improvement in the rep-point image in the presence of high frequency noise when local correlation of edge point directions is required.

A second test was carried out on the images with random data: First, a new object library was constructed using the two sides of the tooth wheel as separate objects. The learning statistics were:

	Reliable	Unique
TOOTHW	48	30
TOOTHDN	30	15

(It should be noted that the two sides of the tooth wheel appear very similar to us despite the high level of UF found by the system). This library was then used to recognize the tooth wheel data with added noise. The recognition results on the TOOTHW with added noise was as follows:

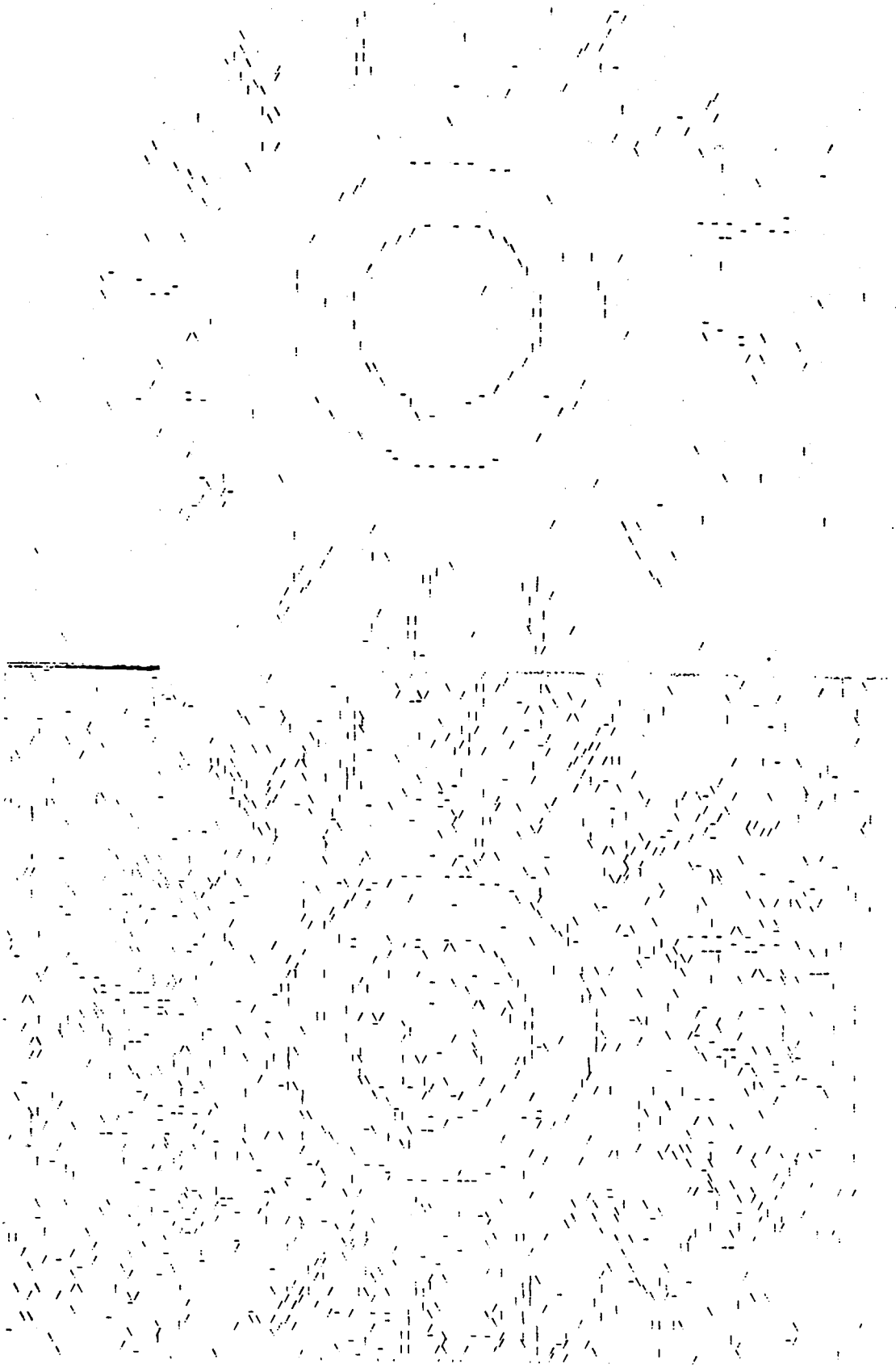


Fig. 5-16 Rep-point image of cutter when SN ratio=4.
Lower image is with no rep-point size threshold

TOOTHW1

σ	S/N ratio	T	TDN
4.03	100	50	0
5.69	50	31	0
9.00	20	34	0
12.73	10	37	1
18.00	5	9	0
20.13	4	7	0
23.24	3	3	0
28.46	2	0	0
40.25	1	3	0

The system did not confuse the two sides of the tooth wheel despite the high level of added noise.

5.2.3. Other Miscellaneous Tests

The system was tested under a variety of other conditions. Most of these tests could not be quantified either due to difficulty in controlling the tests, or because more than a single parameter was changed, or because it was difficult to define a quantitative measure.

5.2.3.1. Object Recognition in Ordinary Lighting Conditions The objects were presented to the system under room lighting conditions (see Fig. 5-17 for the test environment) to verify that it was able to operate despite the non-diffuse and uncontrolled lighting. Fig. 5-18 shows the kind of image produced in this environment. The reader will notice the presence of highlights in the image. It will also be noticed that the lighting was not specially arranged for the vision system (the four tungsten lamps in Fig. 5-17 were used only for the directional lighting test in section {5.2.3.2}; They were normally left switched off), but was what was previously defined as reasonable lighting conditions i.e. lighting designed for human use. The vision

Fig. 5-17 Test environment for operation in 'ordinary' lighting

system is able to operate under these conditions without difficulty. Note that the objects were placed on a grey metal background (see Fig. 5-17). There was also a slight change in scale. Recognition statistics for 6 images of each object in this environment:

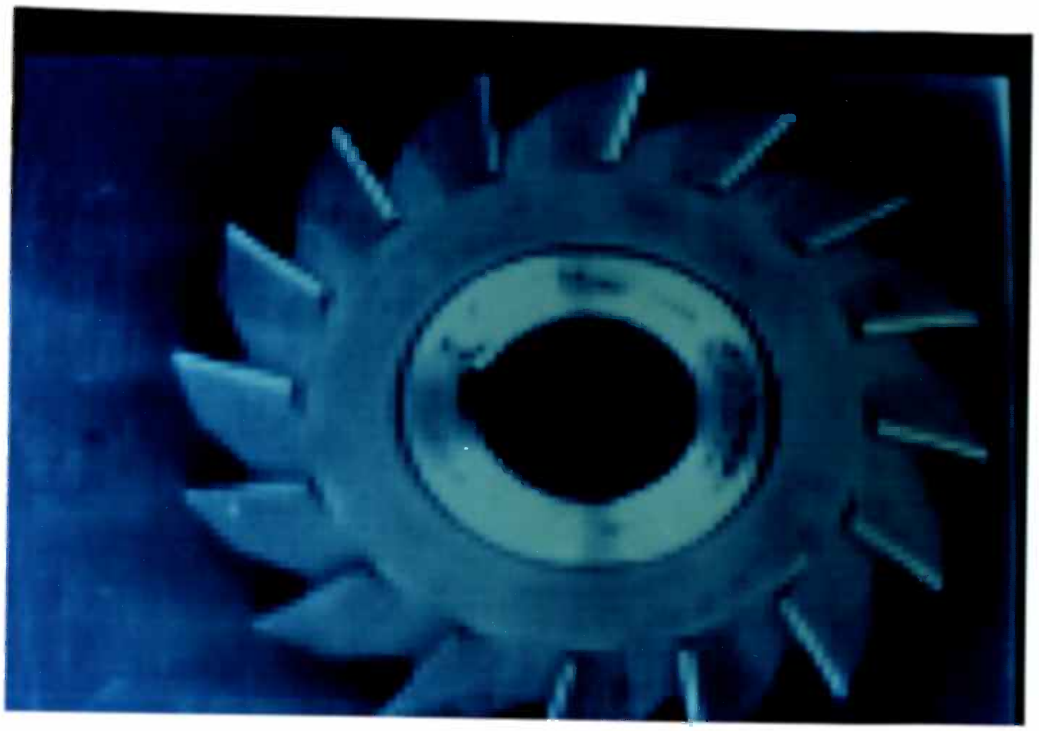


Fig. 5-18 Cutter in ordinary lighting conditions

CUTLAB 1-6

	C	T	G
1	15	0	1
2	17	4	0
3	21	1	1
4	25	0	0
5	16	0	0
6	20	0	1

TOOTHLAB 1-6

	C	T	G
1	0	19	0
2	0	7	0
3	0	25	0
4	0	35	0
5	0	4	0
6	0	6	0

GEARLAB 1-6

	C	T	G
1	0	0	9
2	0	0	11
3	0	0	8
4	0	0	13
5	0	0	8
6	0	0	12

5.2.3.2. Directional Lighting and Camera Blooming

The lights mounted on the roof of the box in Fig. 5-18 were used to introduce directional lighting. Fig. 5-19 shows the sort of image produced. Once again, the system was able to recognize the object using the parts of the object that were not disturbed. The reader will notice the blooming of the CCD image. Recognition statistic for this image:

C	T	G
9	0	3

5.2.3.3. Obscuration Test In this test part of the object was covered by two sheets of white paper so that a sector of the object was visible. The angle of the visible part was reduced until the sys-

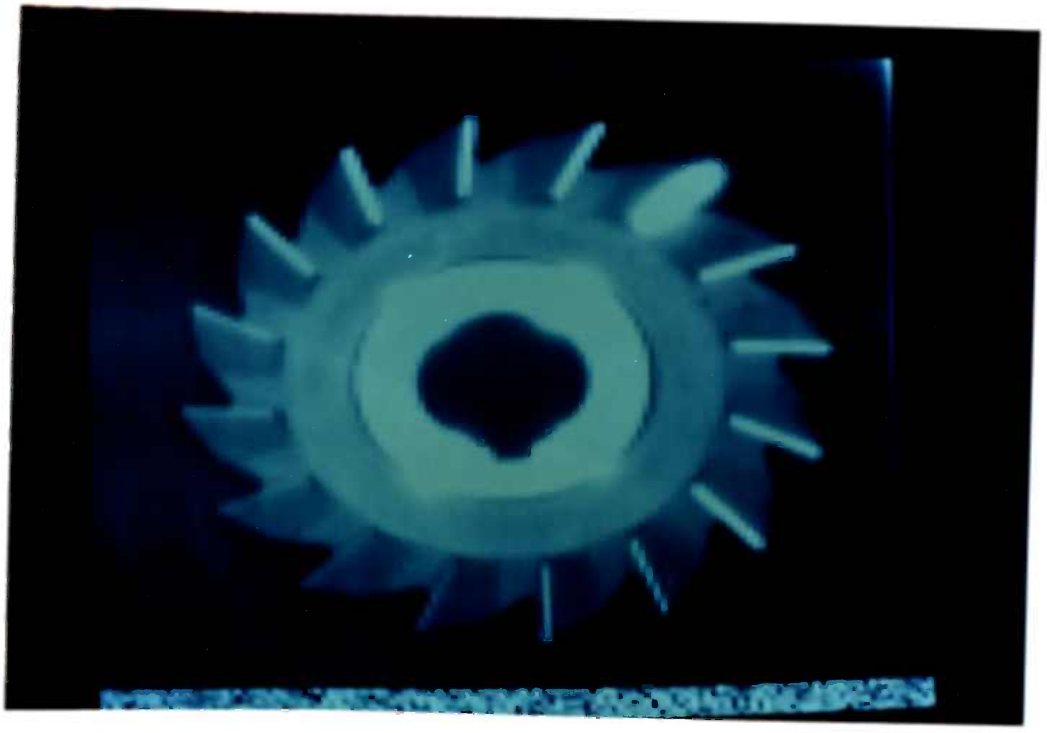


Fig. 5-19 Cutter in directional lighting

tem failed to recognize the object. However, due to certain operational difficulties, this test was not carried out systematically, and I have access only to 4 images of the cutter and tooth wheel. The recognition statistics for these images are unfortunately below the recognition threshold:

CUTOCCL1			TOOTHOCCL1		
C	T	G	C	T	G
3	0	1	0	0	0
3	0	0			
0	0	0			

When these images were acquired, I used a feature match that was based on a subgraph match between neighbourhoods. With this change, the recognition statistic was

CUTOCCL1			TOOTHOCCL1		
C	T	G	C	T	G
30	4	0	2	6	1
21	10	0			
11	3	0			

This method of matching features was discarded due to a higher level of spurious features outside the assumed constancies. Fig. 5-20 shows the tooth wheel when it was recognized by this feature match. Note that the angle of the visible section was 28° of the tooth wheel which is 8% of the surface area. I am unable to present data on the performance of the present system on this test due to inavailability of input data.

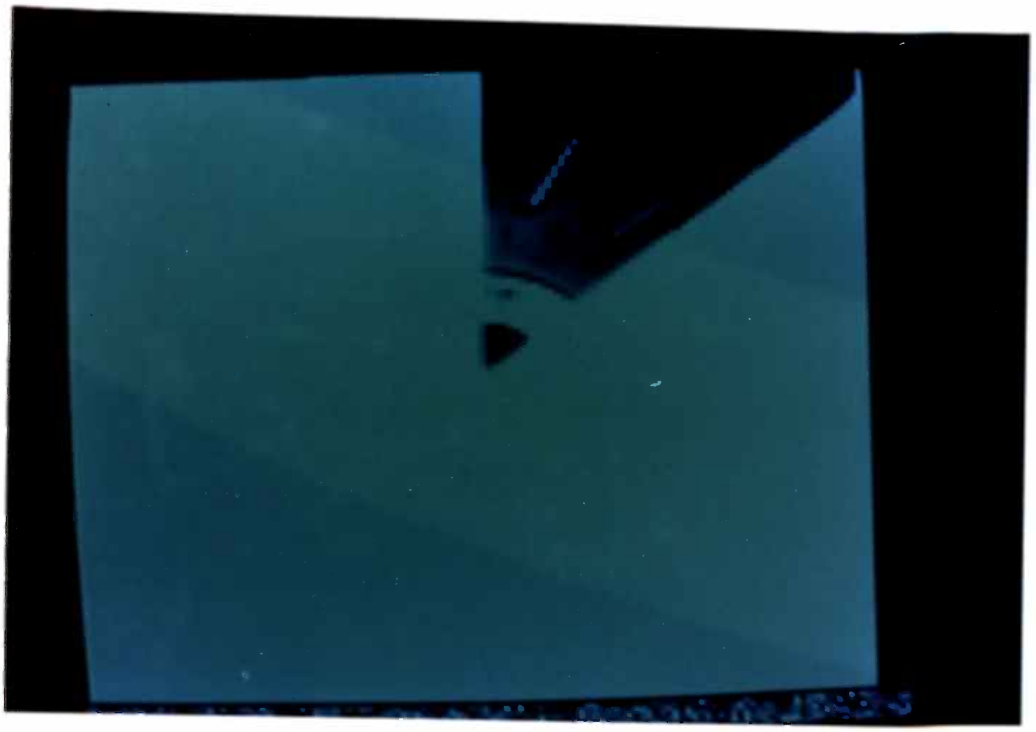


Fig. 5-20 The cutter at the threshold of recognition

5.2.3.4. Distance Variation Test In this test the distance to the object was changed. It should be noted that this changes more than one parameter (i.e. lighting, scale and focus). The system was able to recognize the cutter at least 15cm away from the learned plane (distance to camera 1.2m). Recognition statistics at 15cm:

CUTD 1-2

C	T	G
24	0	3
23	0	6

5.2.3.5. Background Variation In this test the objects were placed on a white sheet of paper and on a chess-board (Fig. 5-21). (A grey background was used in the next test). The recognition statistics were:

CUTW 1-7

C	T	G
8	0	0
6	1	0
13	0	1
7	0	0
12	0	0
16	0	0
9	0	0

CUTCHESS 1-6

C	T	G
15	0	1
22	5	0
8	0	0
23	0	0
28	2	0
10	2	2

TOOTHCHESS 1-6

C	T	G
0	12	0
0	2	0
0	26	0
1	18	0
0	15	0
0	10	0

In order to test the ability of the learning algorithm to learn despite an imperfect background, the system was taught the cutter and the tooth wheel on the chess board using CUTCHESS and TOOTHCHESS. The objective was to determine whether the learning algorithm was able to discard the chess-board features such as the corners of the chess squares. (Note also that the chess-board used had double lines separating the squares). The learning statistics were as follows:

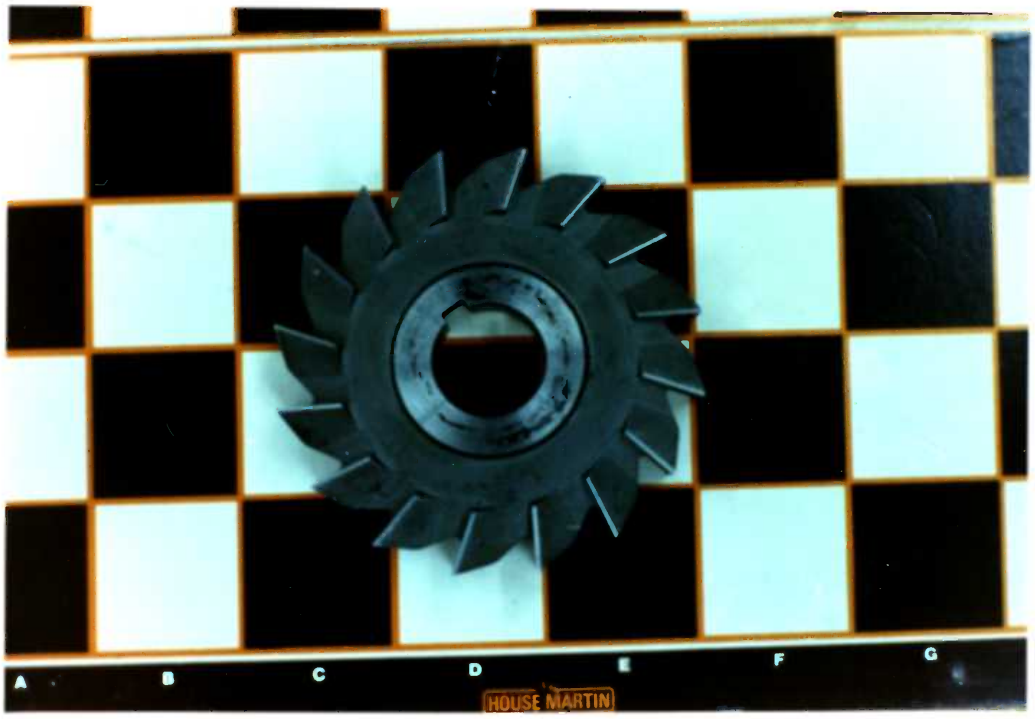


Fig. 5-21 The cutter on a chess board

	Reliable	Unique
CUTCHESS	93	22
TOOTHCHESS	76	3

This library was now used to recognize the cutter and the tooth wheel images that were used in the original CTG library.

CUTTER 1-5

	CUTCHESS	TOOTHCHESS
1	39	0
2	39	0
3	38	0
4	32	0
5	15	0

TOOTHW 1-5

	CUTCHESS	TOOTHCHESS
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0

Note that although the system failed to recognize TOOTHW due to a lack of UF in the learning stage, it did not find any spurious features either. A further test was then carried out to determine the response of the system to the chess-board squares. In this test, the system was taught the cutter and the cutter on a chess board. It was hoped that the chess-board features would then be used as part of the CUTCHESS object, and therefore, the system should classify TOOTHCHESS as being the same, on the basis of the chess board features. The learning statistics are first:

	Reliable	Unique
CUTCHESS	114	39
CUTTER	131	20

Recognition result when TOOTHCHESS was shown:
TOOTHCHESS 1-5

CUTCHESS	CUTTER
297	0
250	0
243	0
301	1
170	0

This demonstrates that the system was able to isolate the chess-board features, which were then used to recognize the chess features in the TOOTHCHESS images. In order to verify that the system was in fact able to differentiate the cutter from the cutter on a chess board, the CUTTER data and the CUTCHESS data were recognized:

CUTCHESS 1-5

CUTCHESS	CUTTER
210	0
97	0
93	0
208	0
166	0

CUTTER 1-5

CUTCHESS	CUTTER
0	50
0	67
0	44
0	39
0	55

(Note that since the library was formed from CUTTER and CUTCHESS, the zero spurious features detected in this table are not significant, as they only serve to confirm the internal consistency expected from an error free learning and recognition program. However, the number of features detected for each of the objects in the image give an indication of the similarity between the selected unique features).

5.2.3.6. Pile of Objects In this test almost all of the parameters were allowed to vary. The objects were presented in a pile on a grey metal background with scratches, in room lighting conditions. See Fig. 5-22 and Fig. 5-23. The system detected 6 UFs of the cutter, 9 UFs of the tooth wheel, and 2 UFs of the gear wheel. (Note: The cutter was placed on the wrong side inadvertently when this image was taken. The recognition statistic was obtained using a library

Fig. 5-22 Pile of objects

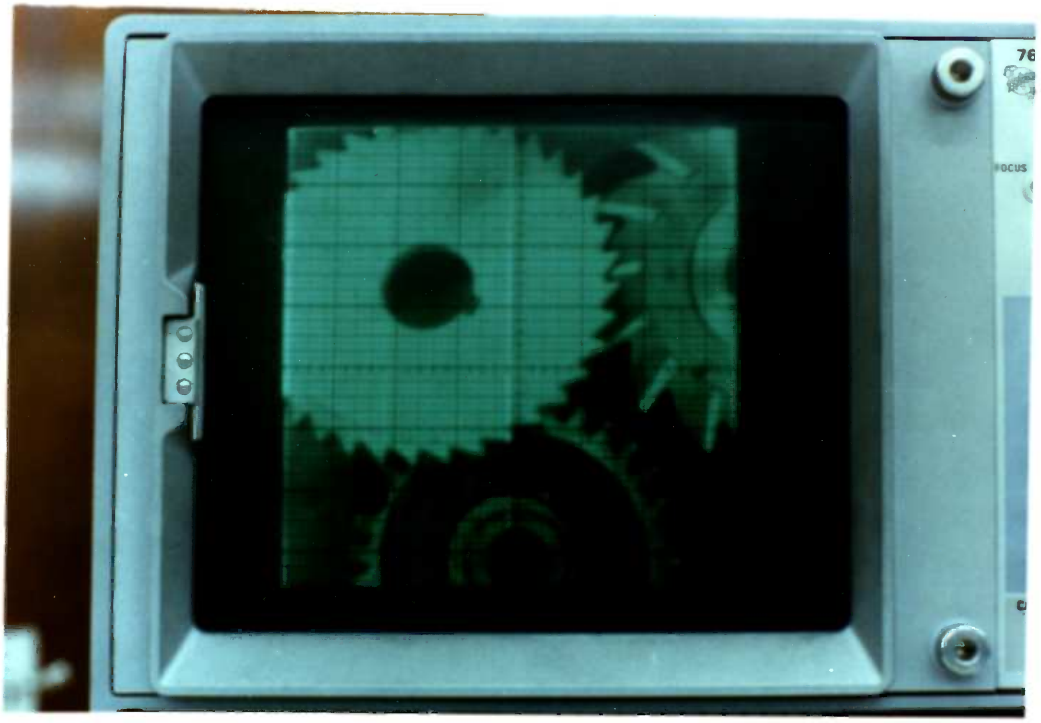


Fig. 5-23 Image of Fig. 5-22

constructed from CUTDN 1-5, TOOTHDN 1-5, and GEARW 1-5, instead of the

CTG library.)

5.2.3.7. Recognition of 'Simple' Objects Although the system is not able to recognize simple objects in principle {section 3.6.5} it should be noted, however, that very complex objects are not necessary for operation. For example, the object in Fig. 5-24 was taught to the system. The learning statistics when this object was compared with the cutter were:

	Reliable	Unique
CUTTER	130	54
SIMPLE	73	46

This is a very high level of unique features for a relatively simple object.

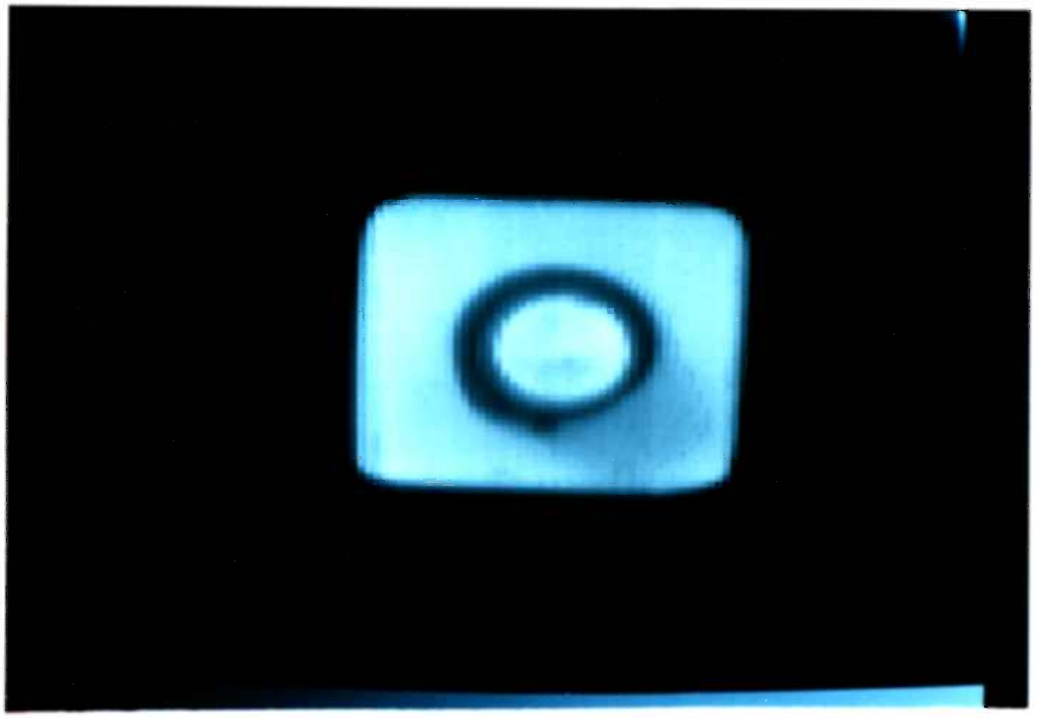


Fig. 5-24 The object referred to as SIMPLE

5.2.3.8. Teaching More Objects In this test, the system was taught 5 objects. The learning statistics were:

	Reliable	Unique	Unique between two				
CUTTER	130	43	-	68	55	50	54
TOOTHW	48	27	28	-	28	30	30
GEARW	18	2	6	11	-	2	11
PULLEY	129	14	51	76	36	-	75
SIMPLE	73	30	46	54	62	34	-

Fig. 5-24 shows the object called SIMPLE and Fig. 5-25 shows the object referred to as PULLEY. Next, two more objects named CUTDN and TOOTHDN were added. These are in fact the opposite faces of the cutter and tooth wheel respectively. From the following learning statistics it will be noticed that the system was able to differentiate the two sides of the cutter and of the tooth wheel without difficulty. The learning statistics were:

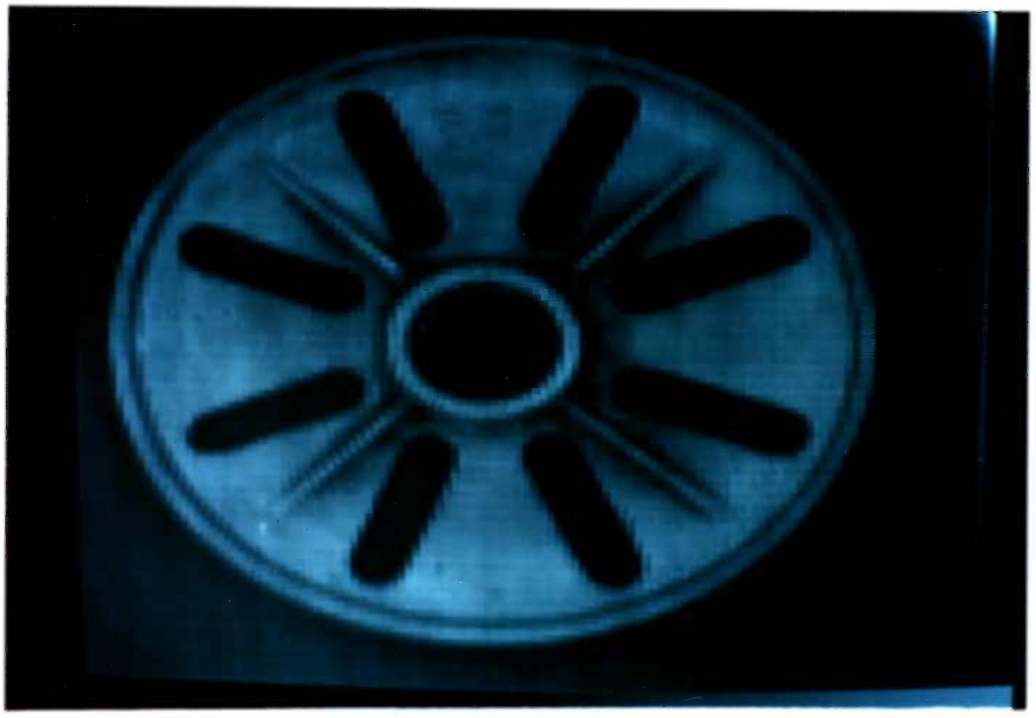


Fig. 5-25 Image of the PULLEY

	Reliable	Unique	Unique between two objects						
CUTTER	130	36	-	68	55	54	50	42	58
TOOTHW	48	26	28	-	28	30	30	28	30
GEARW	18	2	6	11	-	11	2	3	11
SIMPLE	73	16	46	54	62	-	34	27	49
PULLEY	129	11	51	76	36	75	-	54	72
CUTDN	137	44	48	85	60	75	53	-	75
TOOTHDN	30	13	13	15	14	14	14	14	-

(Note: Only 32 UF were used for the CUTTER in the subsequent recognition process.)

It will be noticed that increasing the number of objects does not result in a substantial decrease in the number of UF per object. It should be noted that there is no real need to perform recognition tests on other images as the learning algorithm effectively performs a recognition test over the 5 instances used for each object. However, in order to verify the effect of the new objects on the system, this library was used to re-test the system with the light intensity reduction data. The new statistics for CUTL and TOOTHL were:

CUTL 1-17

	Intensity	CUT	TOO	GEA	SIM	PUL	CDN	TDN
1	4563	44	0	0	0	0	0	0
2	4182	65	0	0	0	1	0	1
3	3950	44	0	0	0	0	0	0
4	3679	60	0	1	0	0	0	0
5	3401	45	0	0	0	0	0	0
6	3088	53	0	0	0	0	0	1
7	2856	41	0	0	0	0	1	1
8	2660	23	0	0	0	1	0	1
9	2381	20	0	0	0	0	0	0
10	2182	14	0	0	0	1	0	1
11	1916	4	0	0	0	1	0	0
12	1630	2	0	0	0	1	0	0
13	1344	1	0	0	0	0	0	0
14	1108	1	0	0	0	0	0	0
15	809	0	0	0	0	0	0	0
16	534	0	0	0	0	0	0	0
17	458	0	0	0	0	0	0	0

TOOTHL 1-31

	Intensity	CUT	TOO	GEA	SIM	PUL	CDN	TDN
1	7349	0	86	0	0	0	0	0
2	6961	0	76	0	0	0	0	0
3	6853	0	46	0	0	0	0	0
4	6490	0	96	0	0	0	0	0
5	6354	0	83	0	0	0	0	0
6	6046	0	85	0	0	0	0	0
7	5823	0	84	0	0	0	0	0
8	5439	0	87	0	0	0	1	2
9	5281	0	108	0	0	0	0	2
10	5003	0	113	0	0	0	1	0
11	4712	0	119	0	0	0	0	1
12	4485	0	117	0	0	0	0	0
13	4279	0	110	0	0	0	1	0
14	3986	0	107	0	0	0	0	0
15	3739	1	132	0	0	0	0	0
16	3435	0	105	0	0	0	0	0
17	3212	0	122	0	0	0	0	0
18	2926	0	125	0	0	0	2	0
19	2635	0	85	0	0	0	0	0
20	2426	0	38	0	0	0	0	1
21	2196	0	44	0	0	0	0	0
22	2009	0	54	0	0	0	0	0
23	1750	0	21	0	0	0	1	0
24	1606	0	21	0	0	0	0	0
25	1494	0	8	0	0	0	0	0
26	1316	0	0	0	0	0	0	0
27	1069	0	0	0	0	0	0	0
28	891	0	0	0	0	0	0	0
29	638	0	0	0	0	0	0	0
30	517	0	0	0	0	0	0	0
31	369	0	0	0	0	0	0	0

The reader will notice the very low number of spurious features and a virtually unchanged resistance to light intensity reduction. An interesting effect of increasing the number of objects has been the reduction in the number of spurious features detected for the gear wheel, when compared with the results for the CTG library in section {5.2.1.1}.

5.2.3.9. Swarf on the Objects Fig. 5-26 is an image of the cutter with swarf thrown over it. The recognition statistic for this image was:

C	T	G
22	0	1

In a further test, a handful of swarf was placed on a grey metal sheet and was then taught to the system as an object. The metal sheet was carefully moved under the camera for the 5 images required so that the swarf did not move relative to the sheet. (See Fig. 5-27). This 'object' was then compared with the cutter to find unique features. The result was:

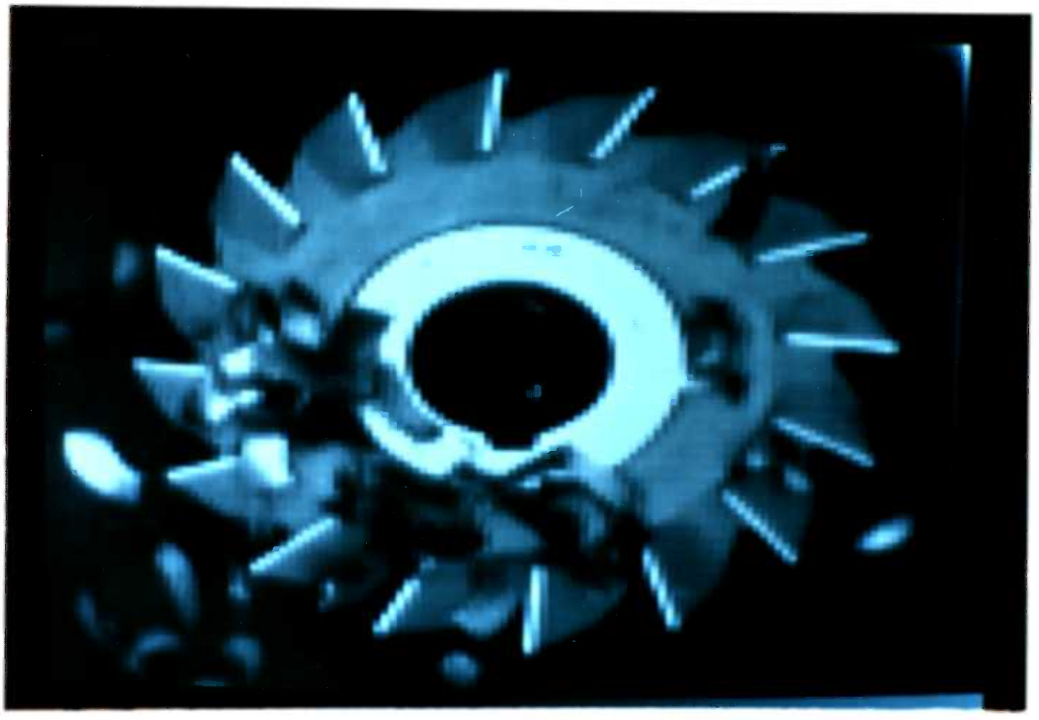


Fig. 5-26 The cutter with swarf

5.2.4. Testing the System Through Variations in Internal Parameters

In these tests the system was tested by changing certain internal operating parameters such as thresholds. The system was found to be insensitive to small variations in threshold values. Some of the system tests described so far were repeated using different thresholds and resulted in similar operating characteristics.

5.2.4.1. Variation of Edge Detector In this test the system was allowed to learn the three test objects (Fig. 5-1) using one of 3 different edge detectors instead of the Walsh transform based edge detector. The edge detectors used were the 2x2 Walsh, 2x2 Roberts, and the 3x3 Sobel operators. The edge detector thresholds were normalized to reflect the size of the detector window. The system was able to find UFs despite variations in the edge detector used.

2x2 Walsh Transform based edge detector

	R	U	Unique between two		
CUTTER	55	17	-	19	17
TOOTHW	66	41	41	-	41
GEARW	30	2	2	11	-

2x2 Roberts edge operator

	R	U	Unique between two		
CUTTER	146	74	-	101	75
TOOTHW	70	35	41	-	35
GEARW	52	16	16	20	-

3x3 Sobel edge operator

	R	U	Unique between two		
CUTTER	132	34	-	57	35
TOOTHW	58	24	26	-	30
GEARW	41	34	30	33	-

(Note: The learning statistics for the Roberts and Sobel operators were obtained after changing one of the thresholds - the minimum number of edge points necessary for a rep-point to be formed - from 2 to 7.)

5.2.4.2. Variation of Local Neighbourhood Radius The objective of this test was to see the effect of varying the local neighbourhood radius on the number of UFs detected by the learning algorithm. Due to limitations in the implementation it was only possible to perform this test for radii from 1 to 9. Fig. 5-28 is a graph showing the variation of the number of UFs found by the learning stage for different values of the radius for the CTG library. The learning statistics were as follows:

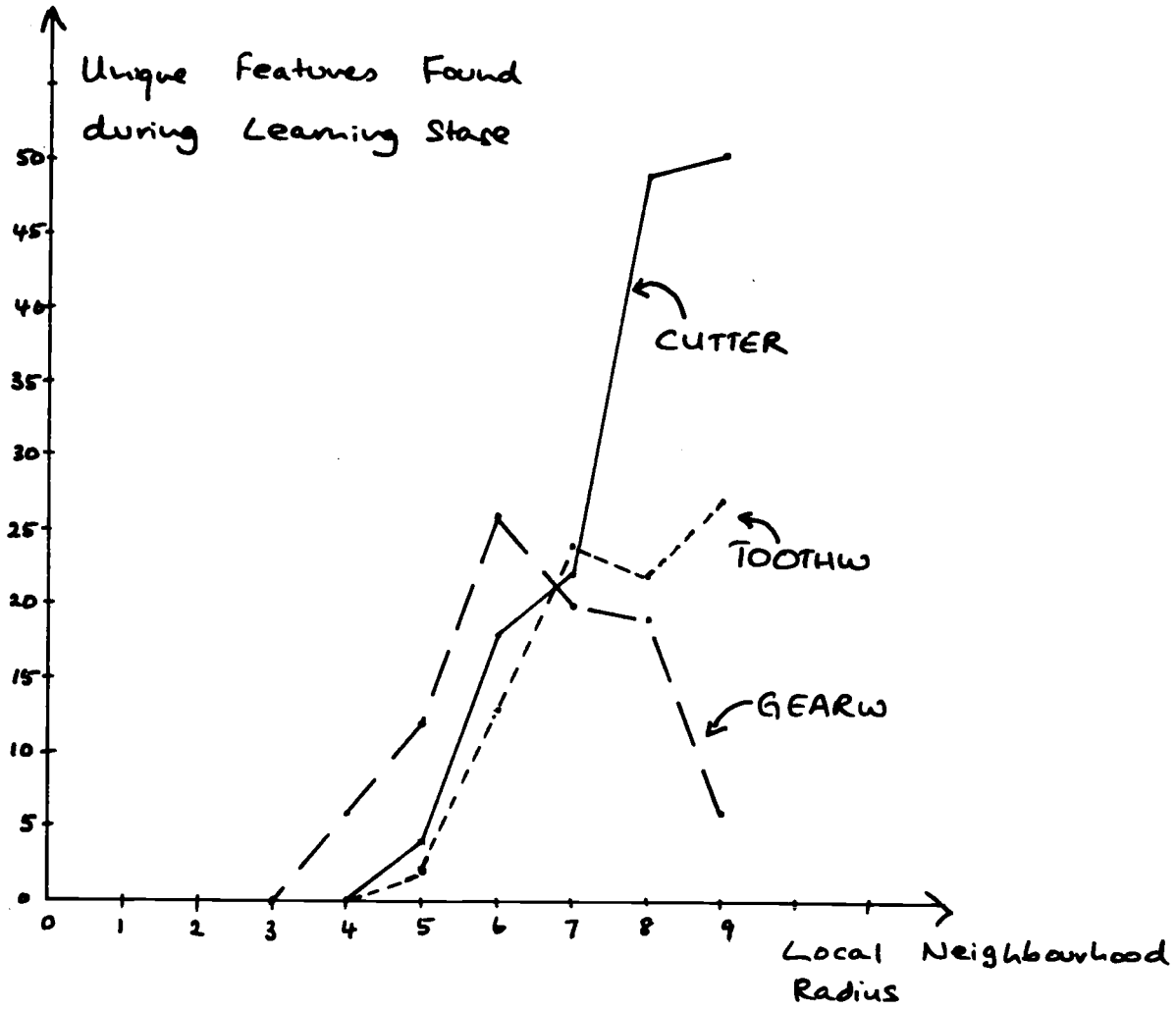


Fig. 5-28 Variation of unique features found during the learning stage

Usual radius=9

Radius=8

	R	U	Unique between two		
CUTTER	132	49	-	84	57
TOOTHW	42	22	31	-	24
GEARW	68	19	23	51	-

Radius=7

	R	U	Unique between two		
CUTTER	140	22	-	87	26
TOOTHW	69	24	44	-	25
GEARW	65	26	28	35	-

Radius=6

	R	U	Unique between two		
CUTTER	123	18	-	66	21
TOOTHW	65	13	39	-	17
GEARW	88	26	41	39	-

Radius=5

	R	U	Unique between two		
CUTTER	75	4	-	37	5
TOOTHW	24	2	9	-	2
GEARW	55	12	31	14	-

Radius=4

	R	U	Unique between two		
CUTTER	8	0	-	7	0
TOOTHW	7	0	1	-	0
GEARW	7	6	7	6	-

Radius=3

	R	U	Unique between two		
CUTTER	0	0	-	0	0
TOOTHW	0	0	0	-	0
GEARW	0	0	0	0	-

5.2.5. Discussion of Overall System Tests

The tests performed on the overall system demonstrate three important points:

(a) The recognition level is above that of chance.

(b) The system performance displays a degree of graceful degradation as the image quality degrades.

(c) The learning algorithm is able to generalize outside the set of sampled images. This shows that the feature descriptor produces features that are structural over a larger range than required by the system constraints, and that the learning algorithm is able to pick these from the rest of the features which are not structural. This is demonstrated by the low level of spurious features detected despite operation outside the sampled domain of the imaging conditions.

5.3. Testing the Individual Parts

These tests were mainly on the pre-processor part of the system. The majority of tests were done to verify the performance of the edge detector and rep-point algorithm.

5.3.1. Testing the Edge Detector

Fig. 5-3 is an edge detected image of the cutter in Fig. 5-7 using the Walsh transform based edge detector (WTED) with the usual thresholds of d=80 and k=0.75. (Note that the edge directions in this figure are quantized to just 4 directions due to display limitations. Computations however are carried out at a resolution of 8 bits.) In the first test, the edge thresholds were allowed to vary. Fig. 5-29 demonstrates the way the edge image changes when the d threshold is changed. (d=160 for (a), d=80 for (b) and d=40 for (c)). Fig. 5-30 shows the variation of the edge image when the k threshold is changed. (k=0.9 for (a), k=0.75 for (b), k=0.6 for (c), and k=0.0001 for (d)). No edge points were detected when k=1.0. Note the explosion in the number of edge points (and line thickening) as k is reduced. (See graphs in Fig. 5-31 and Fig. 5-32).

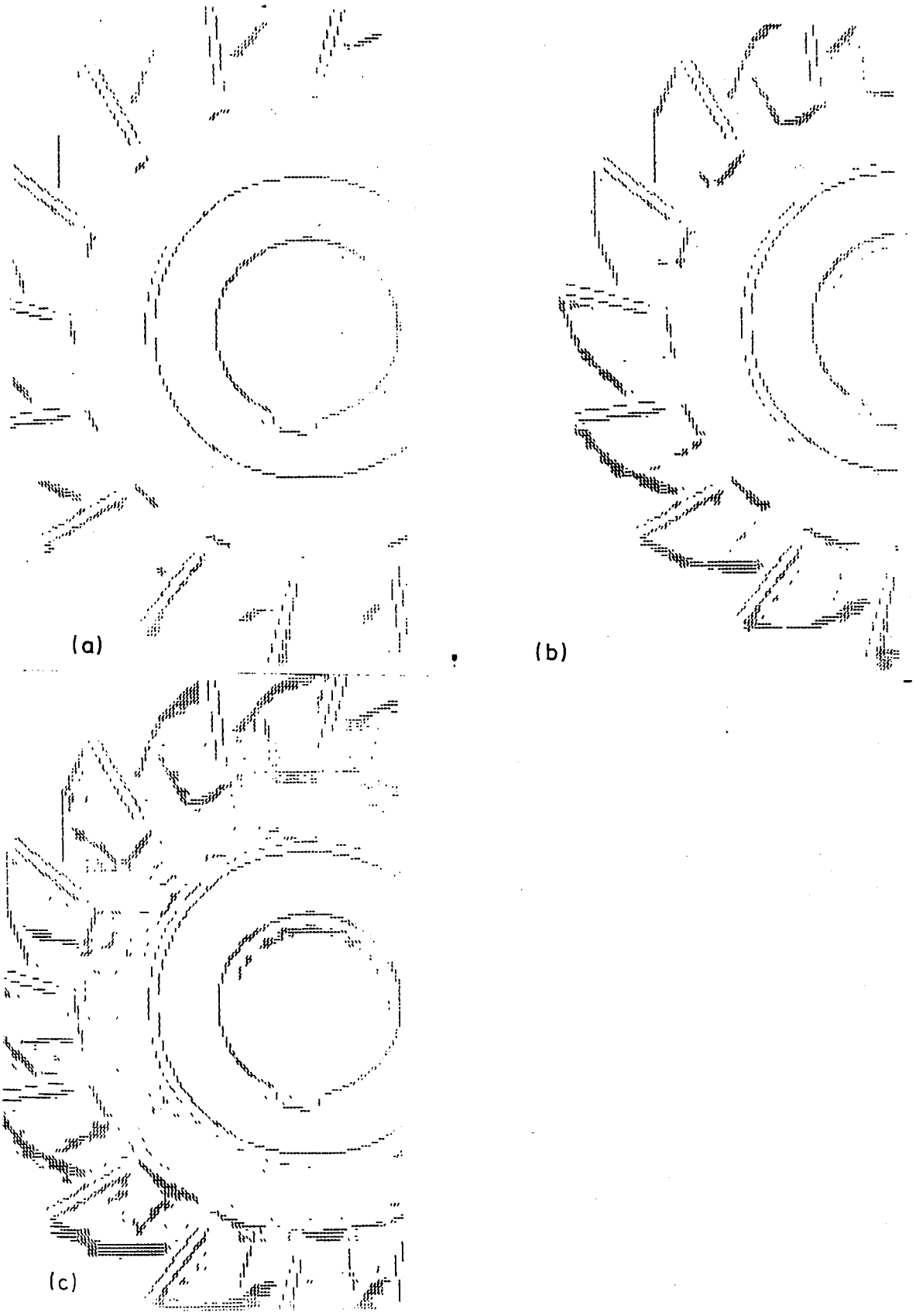


Fig. 5-29 Variation of edge image with d thresho

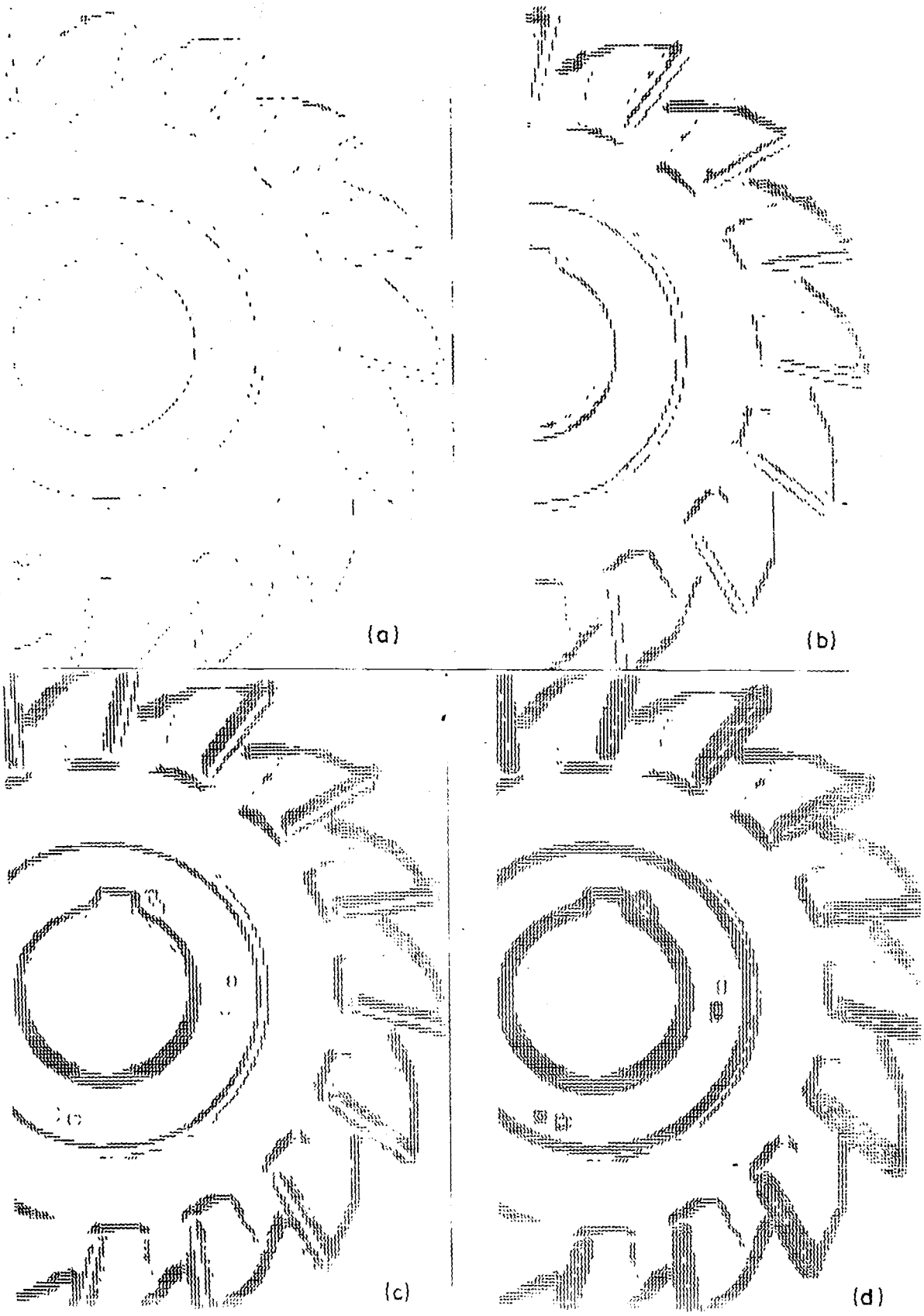


Fig. 5-30 Variation of edge image with k threshold

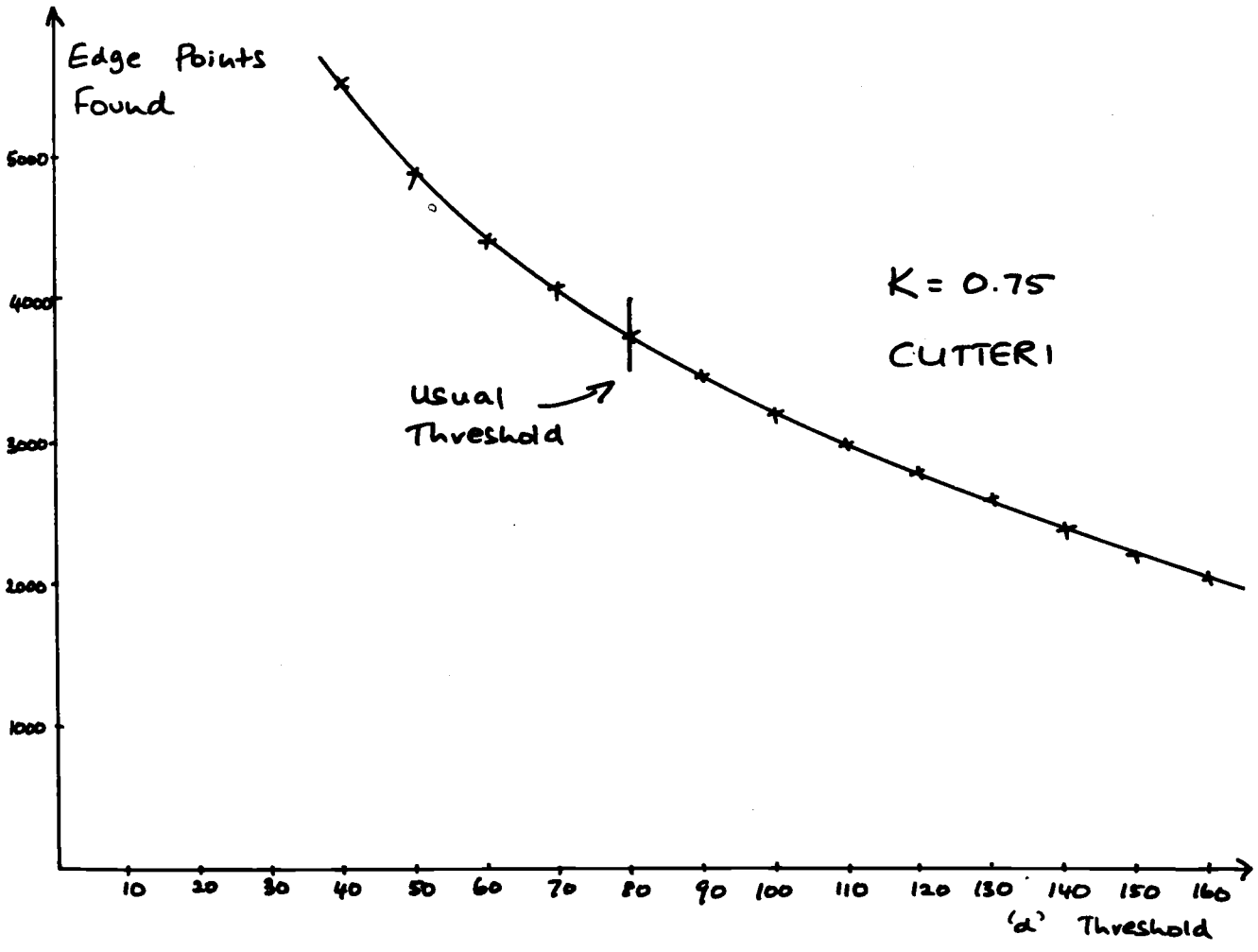


Fig. 5-31 Variation of number of edge points with d threshold

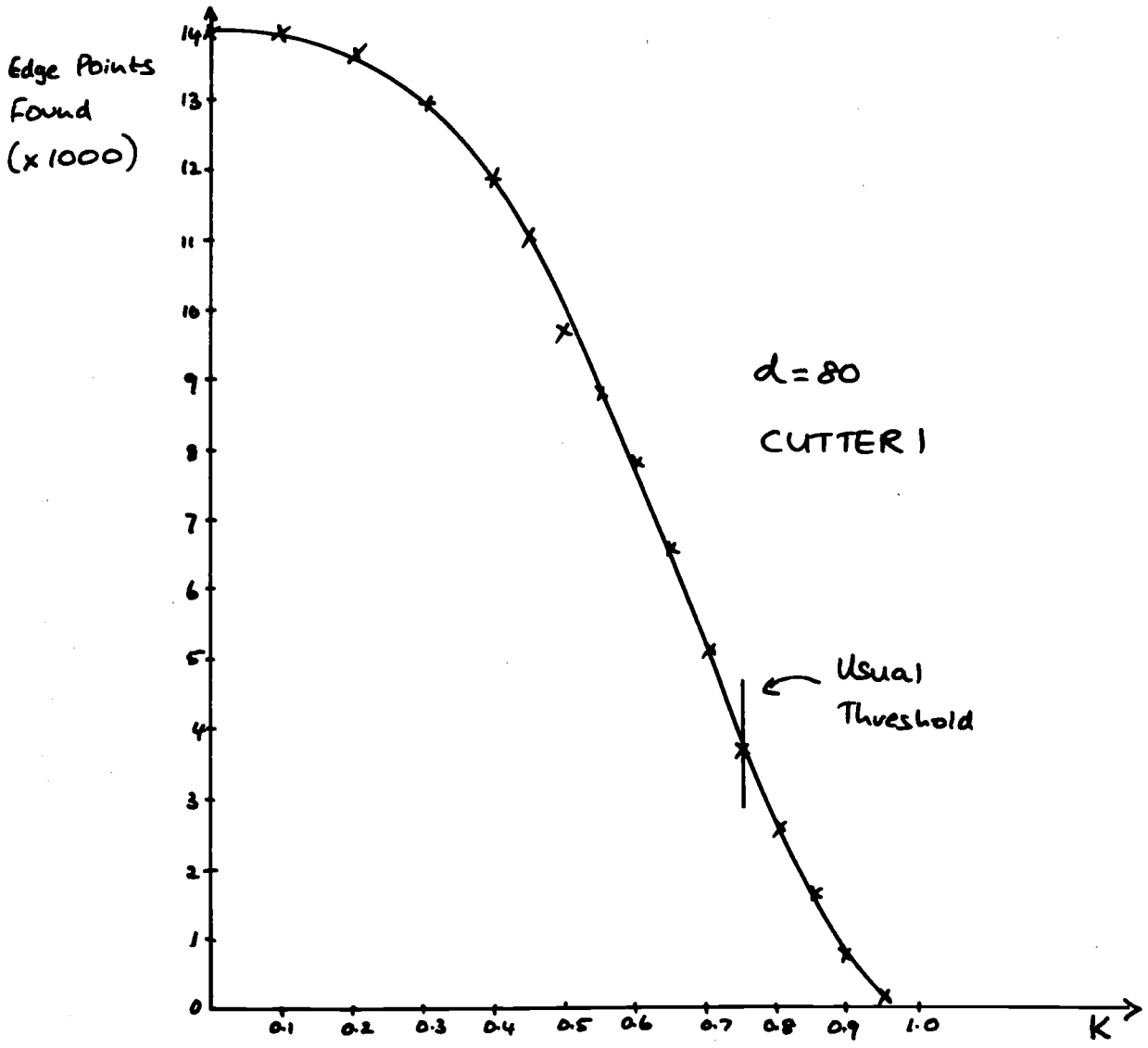


Fig. 5-32 Variation of number of edge points with k threshold

The importance of the k threshold becomes evident when the WTED is compared with other edge detectors. Fig. 5-33 shows the edge image produced by three other edge detectors. The edge thresholds were normalized to match the detector window size. ((a) Roberts 2x2, (b) Sobel 3x3, (c) Walsh 2x2, (d) Walsh 4x4). The Roberts and Sobel edge detectors result in much thicker edges. The performance of these two are equivalent to the performance of the Walsh edge detector when k=0. The following is a list of the number of edge points detected by the different edge detectors in Fig. 5-33.

Edge detector	Number of edge points
Roberts 2x2	10898
Sobel 3x3	13593
Walsh 2x2	6924
Walsh 4x4	3741

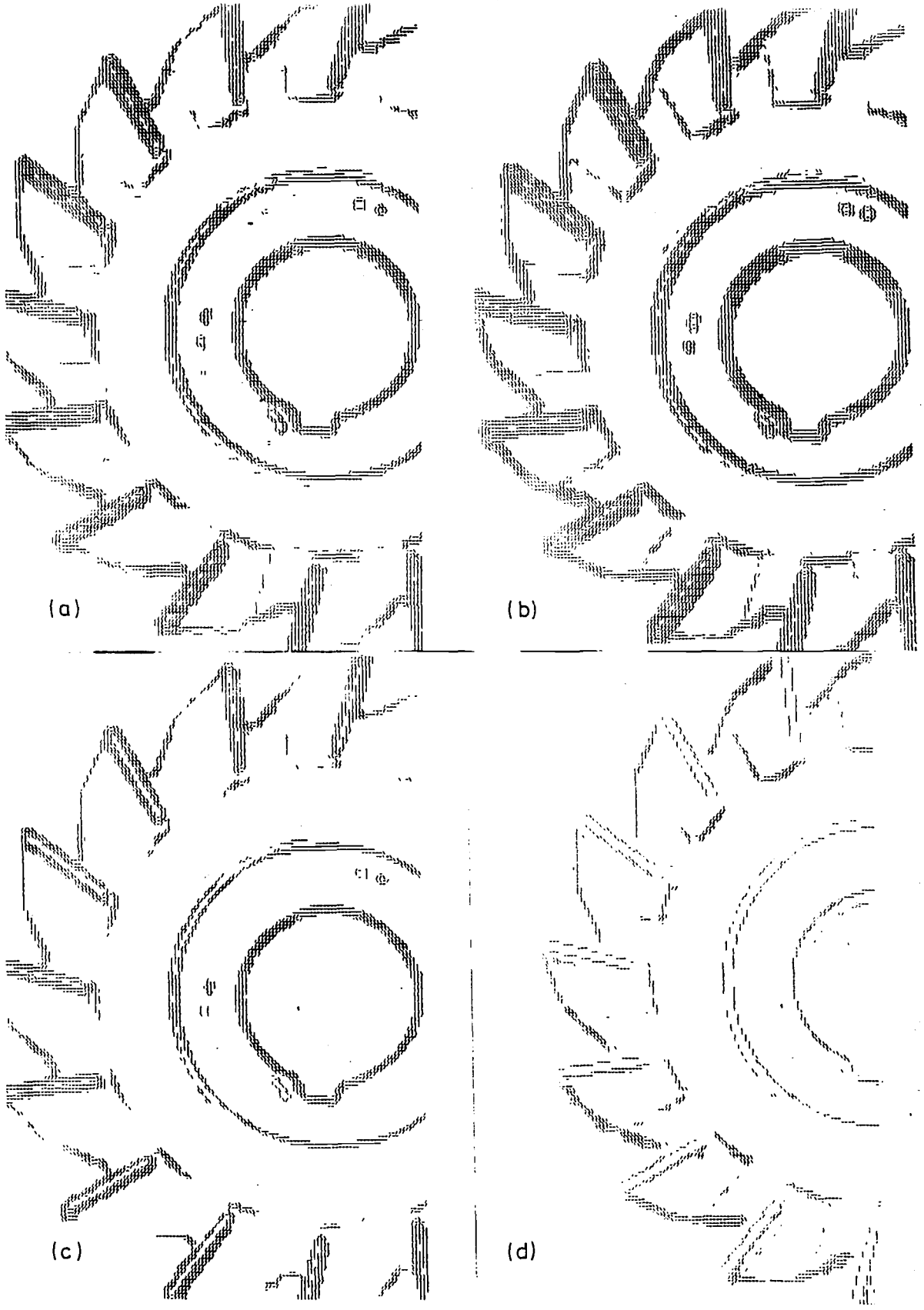


Fig. 5-33 Edge image using 4 different edge detectors

5.3.1.1. Noise performance of the WTED In order to verify the response of the WTED to random noise, an ideal step edge with added Gaussian noise was used. The Pratt [1978] p.497 goodness measure was used to evaluate the performance of the edge detector. This measure was defined to be R where

$$R = \frac{1}{I_I} \sum_{i=1}^{I_A} \frac{1}{1+\alpha d^2}$$

where $I_I=\max(I_I,I_A)$ and I_I and I_A represent the number of ideal and actual edge points, α is a scaling constant, and d is the separation distance of an actual edge point from an ideal edge.

The step edge size used was 20 and the average brightness was 64. The image size was 128x128. WTED thresholds were d=80 and k=0.75. Fig. 5-34 (a)-(d) shows the edge image for signal to noise ratios of ∞, 400, 100, and 25. Fig. 5-35 (e)-(f) shows the edge image for S/N ratios of 16, 4, and 1. The goodness measures were as follows:

(These figures are for a step size of 25).

S	S/N ratio	Goodness
2.5	100	100.0%
3.53	50	99.2%
5.59	20	82.7%
7.90	10	35.5%
11.10	5	16.6%
12.50	4	15.2%
14.40	3	13.5%
17.68	2	10.6%
25.00	1	9.3%

Fig. 5-36 is a graph of this variation. Note that the Pratt goodness measure does not use edge direction data.

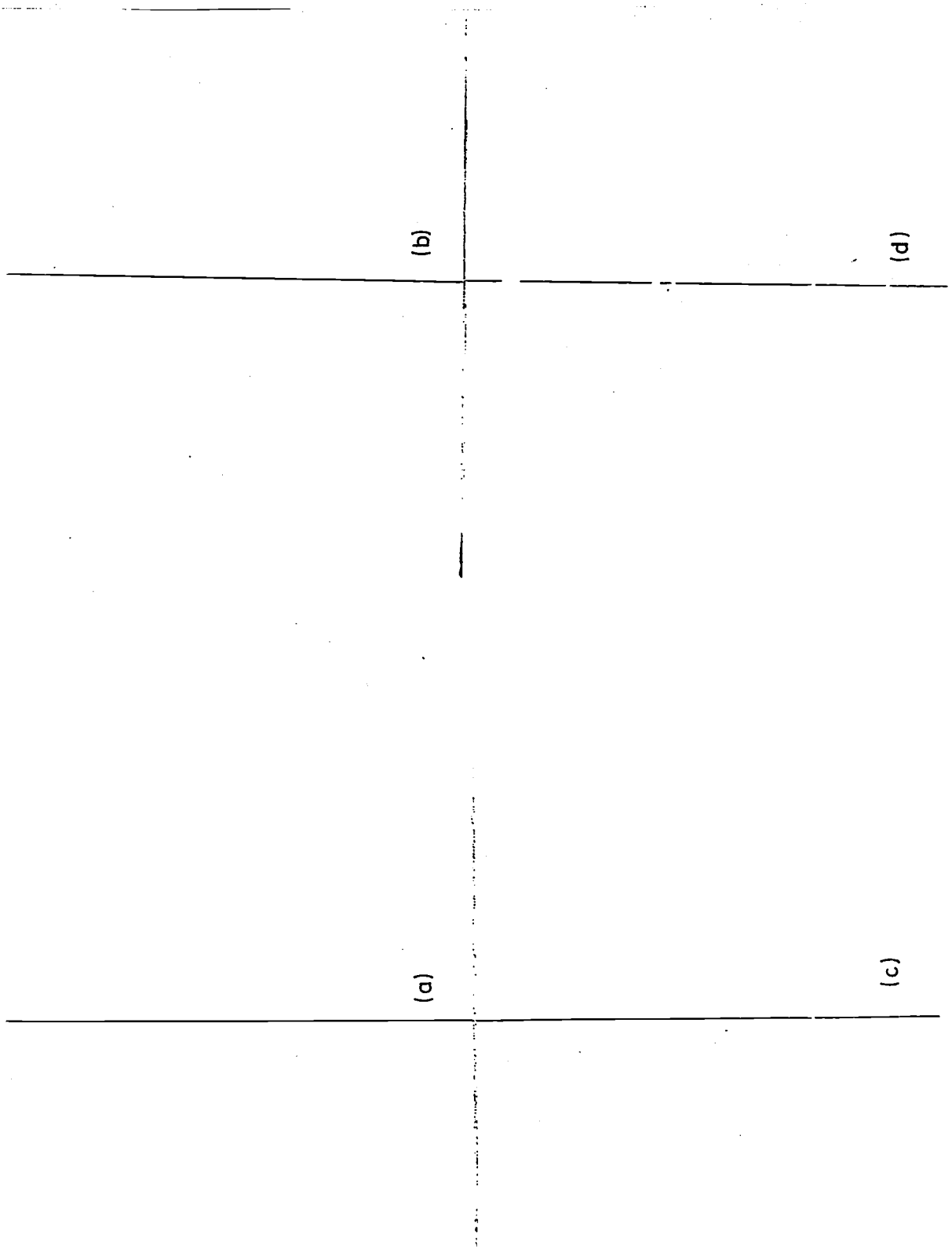


Fig. 5-34 Edge images for S/N ratios of ∞, 400, 100, and 25

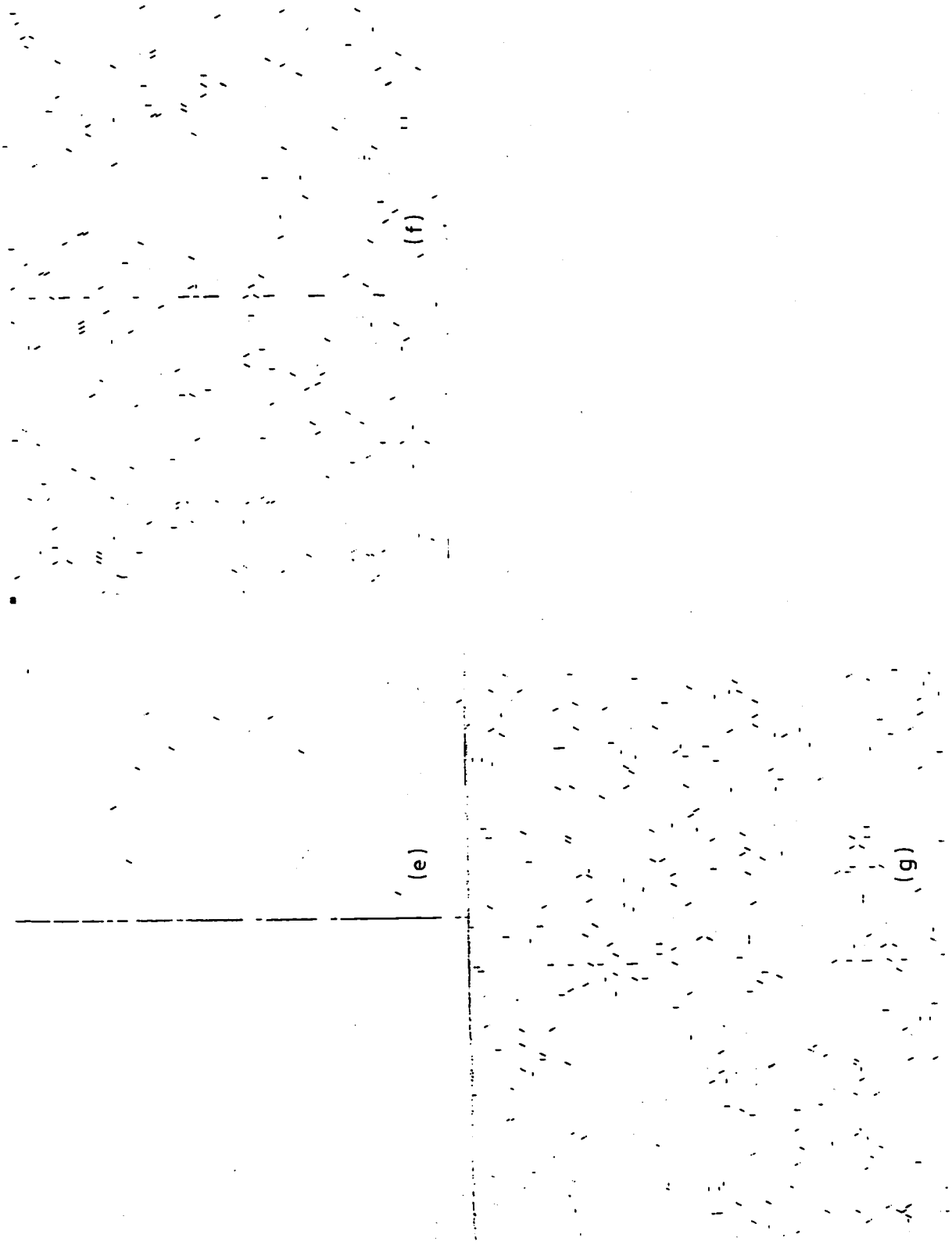


Fig. 5-35 Edge images for S/N ratios of 16, 4 and 1

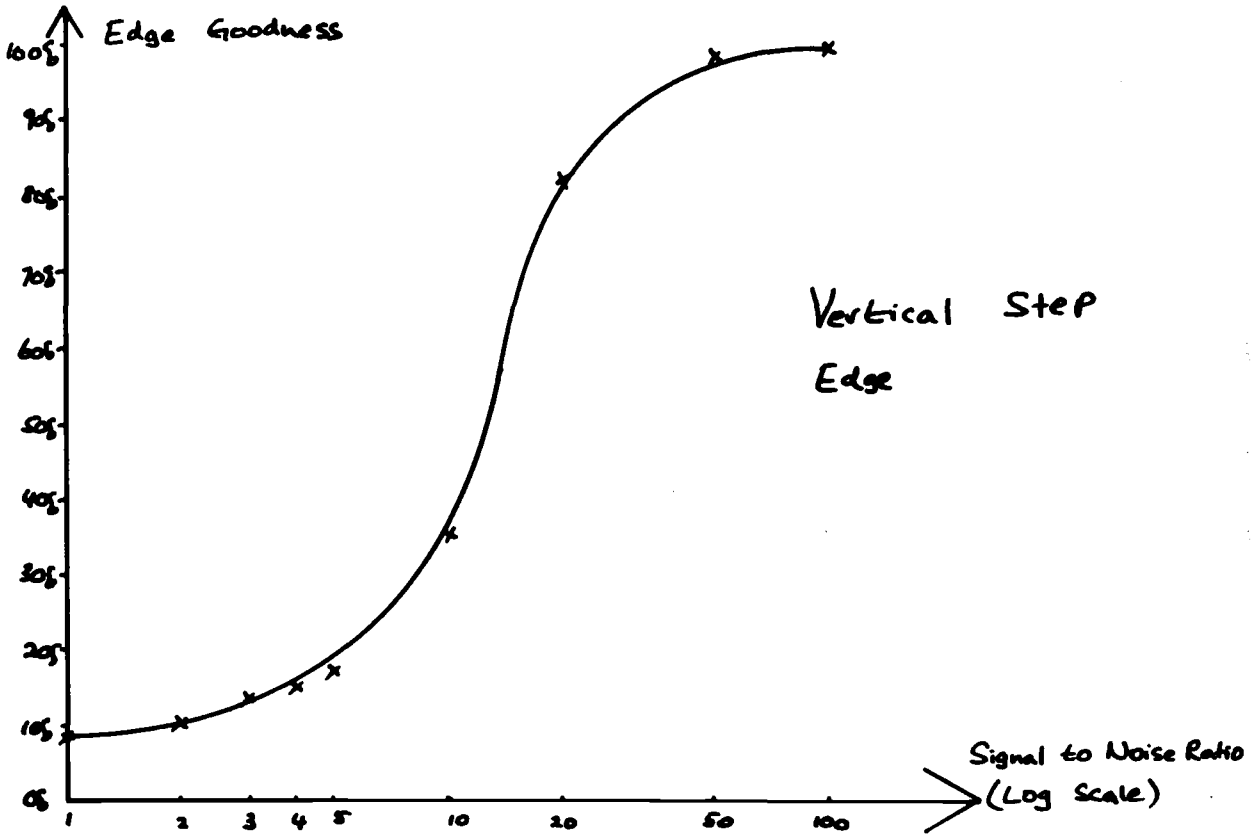


Fig. 5-36 Variation of Pratt goodness measure with added noise

5.3.2. The Rep-point Algorithm Fig. 5-4 is a rep-point image of Fig. 5-3. Fig. 5-37 and Fig. 5-38 show the rep-point images produced when the edge detector thresholds were changed. The threshold values were as follows:

For Fig. 5-37, (a) d=40, (b) d=60, (c) d=100, (d) d=120, k=0.75.

For Fig. 5-38, (a) k=.001, (b) k=.4, (c) k=.6, (d) k=.9, d=80.

Fig. 5-39 and Fig. 5-40 are graphs of the variation of the number of rep-points. The Pratt goodness measure was slightly modified so that it could be used on rep-point images. The goodness measure when Gaussian noise was added was as follows:

(These figures are for a step size of 25).

S	S/N ratio	Goodness
2.5	100	100.0%
3.53	50	100.0%
5.59	20	95.2%
7.90	10	80.9%
11.10	5	44.0%
12.50	4	37.5%
14.40	3	37.7%
17.68	2	24.1%
25.00	1	19.6%

Note that the rep-point image goodness measure is better than that of the edge detector. (See graph in Fig. 5-41).

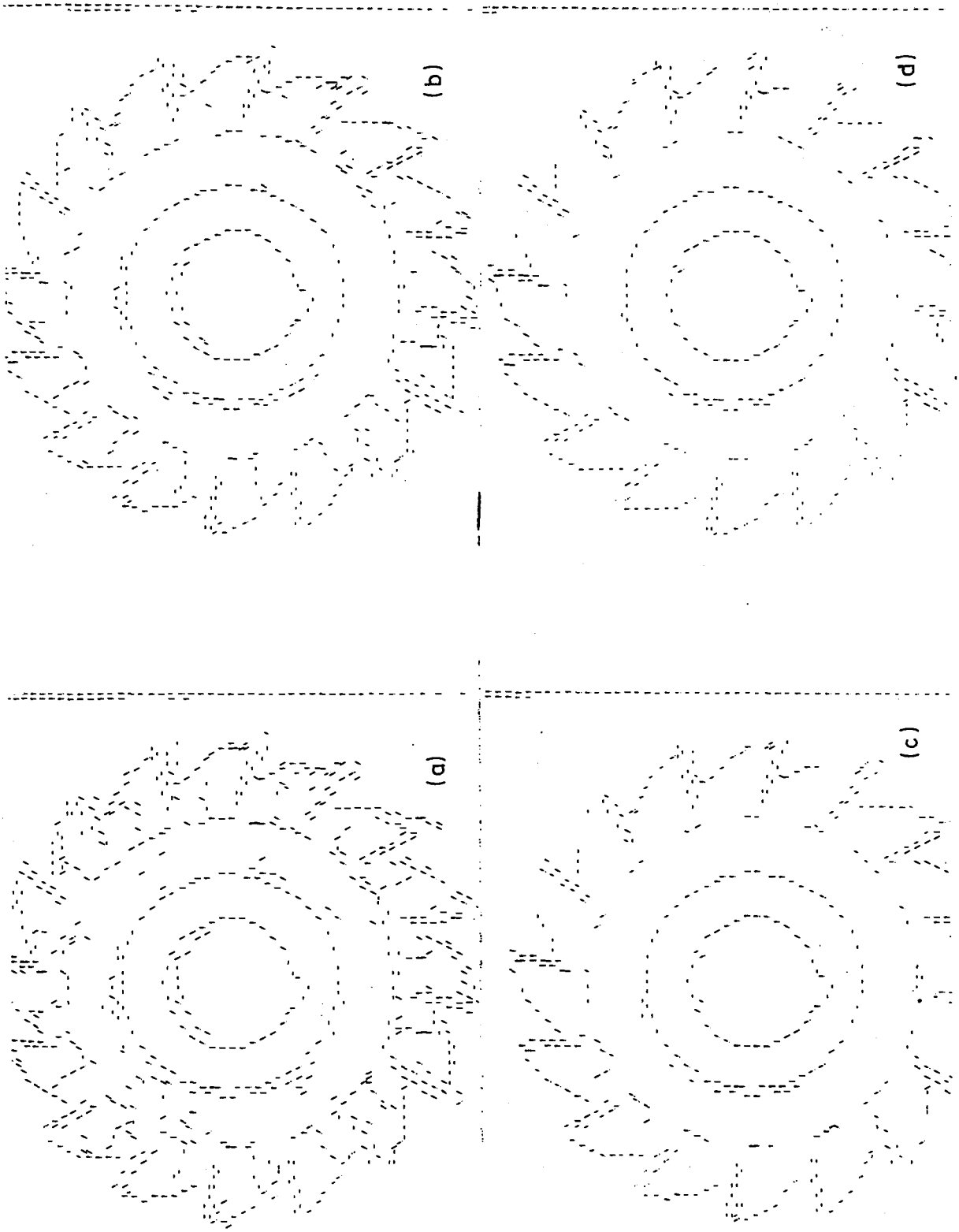


Fig. 5-37 Variation of rep-point image with d threshold

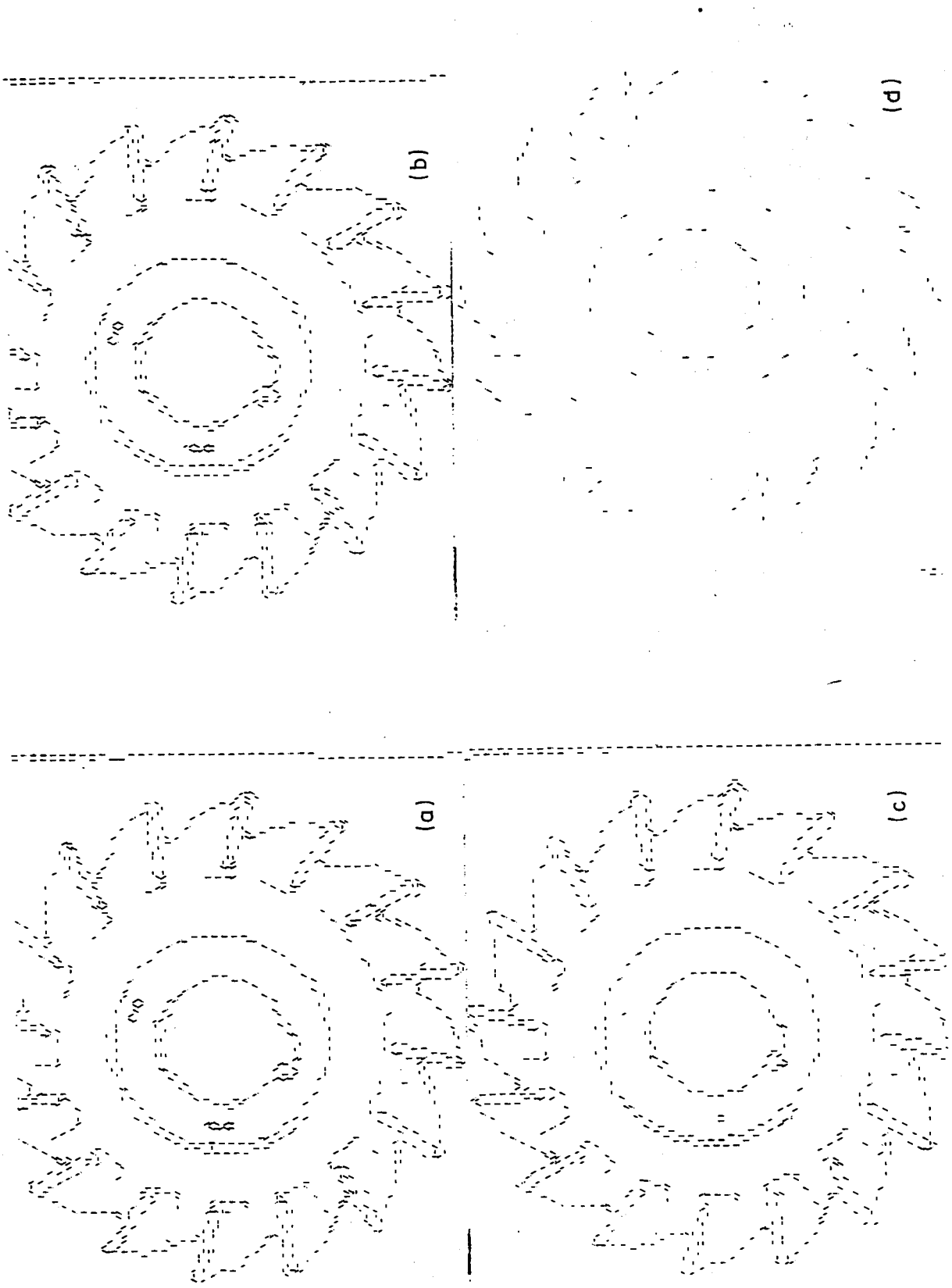


Fig. 5-38 Variation of rep-point image with k threshold

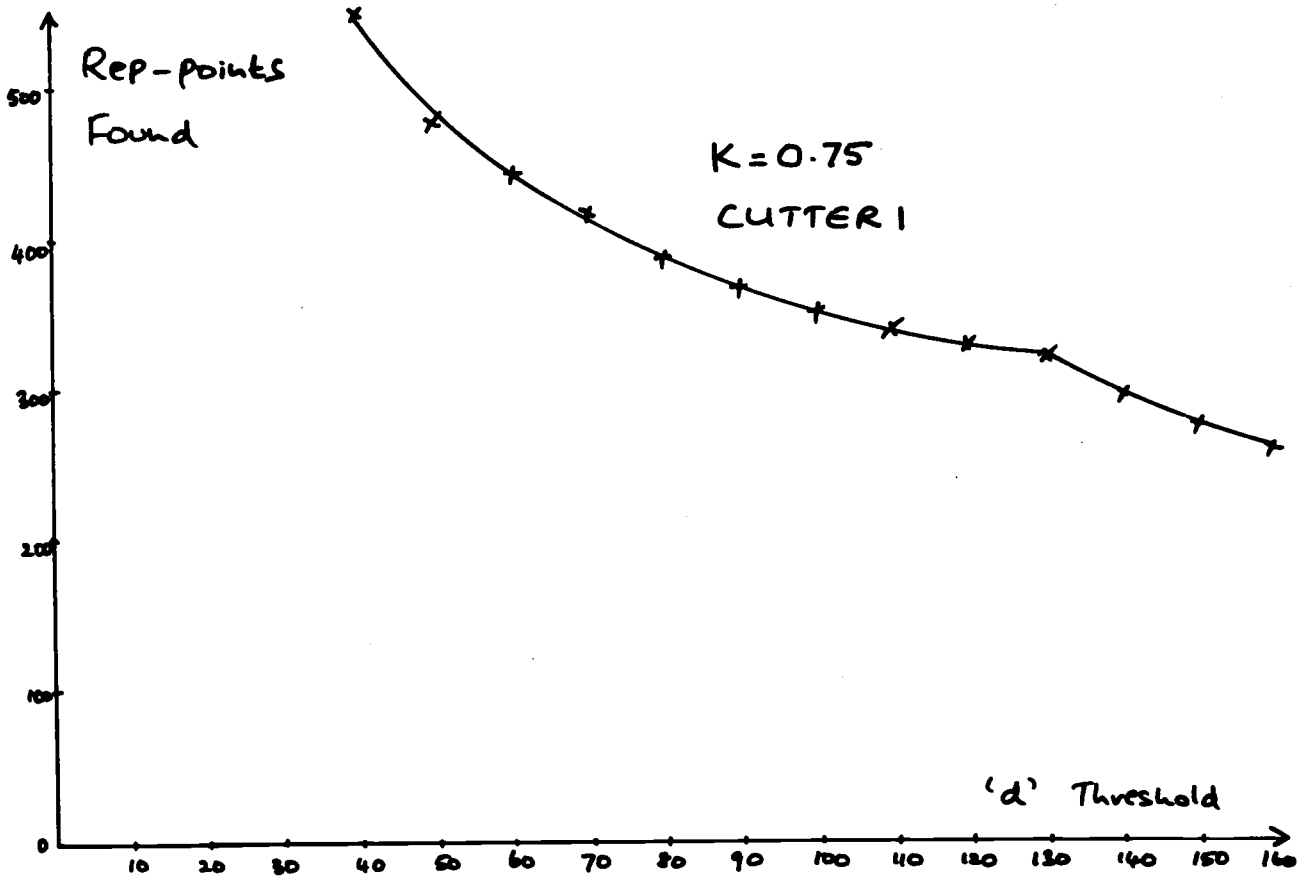


Fig. 5-39 Variation of number of rep-points with d threshold

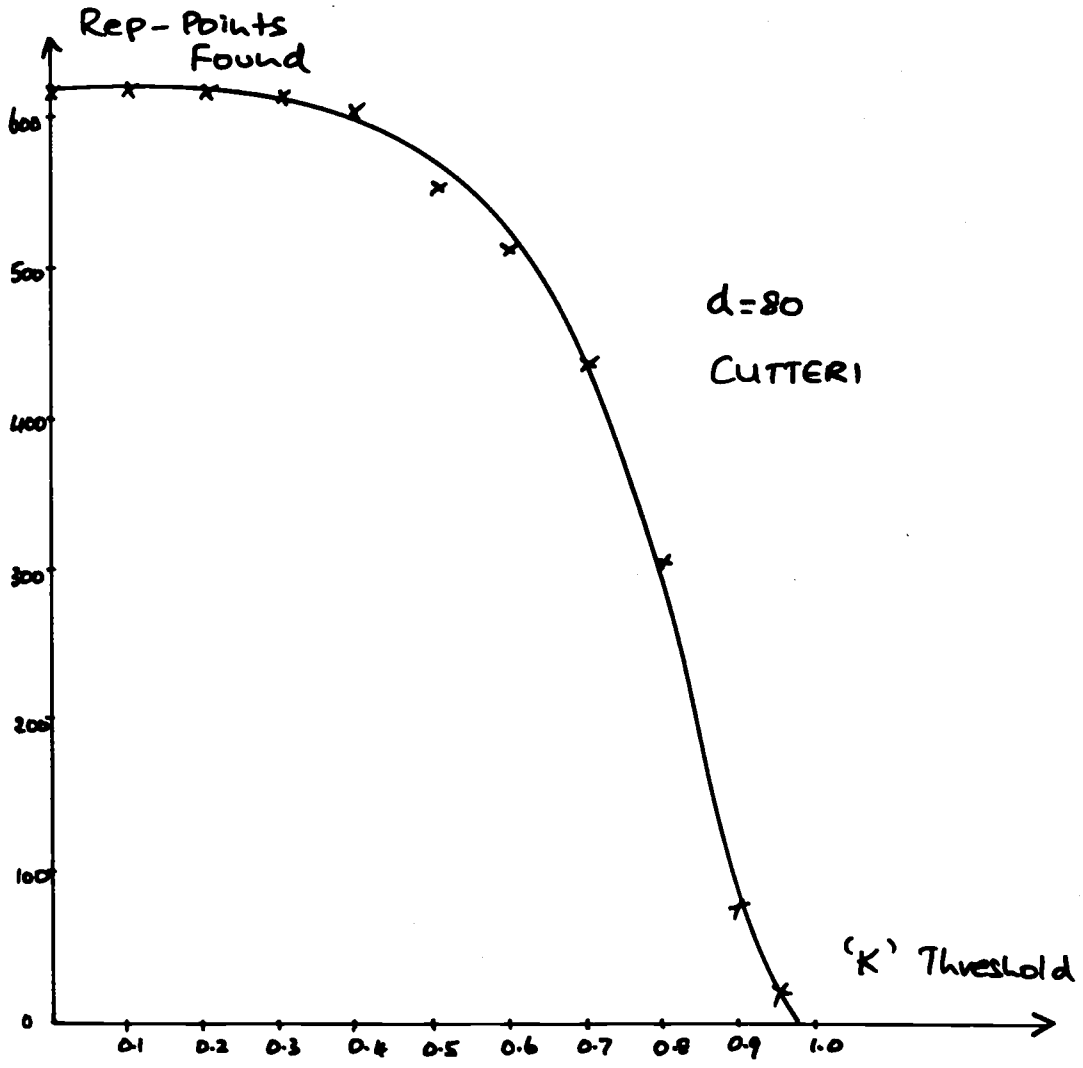


Fig. 5-40 Variation of number of rep-points with k threshold

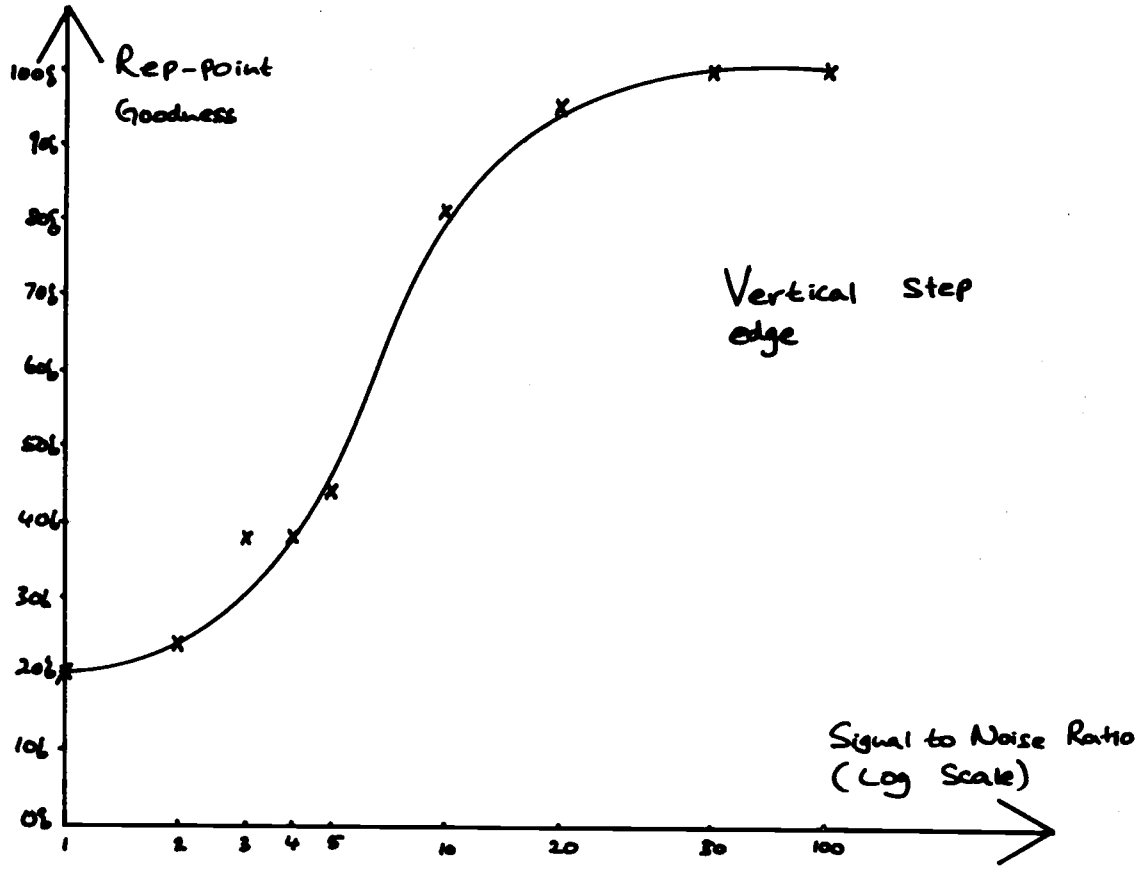


Fig. 5-41 Variation of modified Pratt goodness measure for rep-points

5.3.3. Local Neighbourhood Statistics

Following are histograms of the number of rep-points in a local neighbourhood for 2 instances each of the cutter, tooth wheel and gear wheel:

Rep-points	CUTTER1	CUTTER2	TOOTHW1	TOOTHW2	GEARW1	GEARW2
1	3	4	2	1	3	0
2	42	45	12	13	2	7
3	81	71	42	36	26	22
4	101	95	54	51	54	41
5	44	60	62	47	63	57
6	56	53	37	66	54	60
7	34	59	43	51	69	47
8	30	29	33	31	44	59
9	24	16	16	16	40	35
10	7	11	13	2	20	30
11	2	5	10		10	12
12	2		5		4	2
13			2			1

The following table shows the way the local neighbourhood rep-point count histogram varies when the radius of the local neighbourhood is changed, for one object instance of the cutter.

CUTTER1

Radius	Number of rep-points											
	1	2	3	4	5	6	7	8	9	10	11	12
8	3	42	81	101	44	56	34	30	24	7	2	2
7	7	80	99	62	55	51	19	9	6			
6	15	109	95	65	42	30	8					
5	42	127	82	60	29	8	1					
4	82	124	50	45	12							
3	110	56	14	2								
2	45	9										
1	1											

5.4. Execution Time

5.4.1. The Pre-Processor

The total pre-processing time for a single 256x256 image on the PDP11/24 is about 72s excluding time needed for disc I/O between individual pre-processor stages. (Disc I/O time is not included as it is not an essential part of the pre-processor). Of this time, the edge detector consumes approximately 55s, the rep-point algorithm about 14s, and the local neighbourhood algorithm about 1-3s. However, it is expected that any industrial implementation would use dedicated hardware to reduce these times to a negligible level. A hardware architecture for doing so is presented in chapter 6.

5.4.2. The learning stage

The learning stage requires approximately 2-5 minutes to construct the unique data structures from the object instance models. This is for 3 learned objects. As described in section {4.3.2} the learning time is proportional to the square of the number of objects in the worst case. However, the learning time is assumed to be non-critical. If it were necessary to change the object library rapidly, one possibility is to construct all of the libraries in advance, and hold them on disc. It would for instance be possible to set up libraries based on different subsets of the learned objects in order to respond to particular needs dynamically.

5.4.3. Recognition Time

The time required to search for all unique features varies from 1s to 5s when searching for 3 objects (the CTG library). However, there is never a need for searching for more than 6 UF for any object, as a 100% confidence level has been reached by this stage (although it is necessary to search for UF of the other objects which have not been recognized). Thus, it is possible to use a variety of heuristics to limit the execution time depending on the expected operating conditions etc.

For example, when ideal operating conditions are guaranteed, and only a single object is known to be present in the image, the recognition time is about 0.1-0.5s (when 3 objects are being searched for). In special circumstances (when only a single UF is sufficient for recognition, and a small object is in the image), execution times as low as 10ms have been observed (for a two object library).

Clearly though, execution times are highly dependent on the hardware that is used. The reported times were obtained on a PDP11/24 minicomputer in Fortran. It is estimated that a speed improvement of a factor of 10 could be obtained if the recognition algorithm was implemented in assembly code on a 68020 microprocessor operating at 20MHz. A further improvement could be achieved by removing the need to unpack rep-point data {section 4.1.3}. At present, the recognition algorithm spends much time doing so.

5.5. Summary of test results

The system was tested by varying a single imaging parameter until recognition failed. This allowed us to map the vision system sensitivity to each imaging parameter. The results showed that the system performance degraded gracefully, so that catastrophic failure was not observed in any of the tests.

Sensitivity to three main assumptions

- Light intensity could be reduced upto 80%. Lowering the edge threshold improved the performance further. {section 5.2.1.1}.
- Object scale could be reduced upto 43%. {section 5.2.1.2}.
- The plane of the object could be tilted away from the learned 2D plane (by placing the object on an inclined plane) by upto 47°. {section 5.2.1.3}.

Sensitivity to implicit assumptions

- A degree of image blurring could be tolerated. {section 5.2.2.1}.
- Gaussian noise could be added until the signal to noise ratio was as low as 6.0dB. {section 5.2.2.2}.

Other tests

- The system could cope with non-ideal lighting. {section 5.2.3.1}.
- Directional lighting and camera blooming did not destroy recognition. {section 5.2.3.2}.
- Objects could be recognized despite partial obscuration by a sheet of paper. {section 5.2.3.3}.

- Distance between the object and the camera could be changed (despite a scale change, lighting change, and blurring). {section 5.2.3.4}.
- The image background could be changed to be a low contrast grey metallic background, a white sheet of paper or a chess board. {section 5.2.3.5}.
- Two objects from a pile of objects was recognized despite scale changes, lighting variations, low contrast background, object obscuration, and rotation of object plane away from learned 2D plane. {section 5.2.3.6}.
- The object library could be changed by adding other objects, including the reverse side of the cutter and toothwheel, and objects with simpler shapes. Increasing the number of objects in the known library (to 7) in fact resulted in an improvement in performance in terms of detected spurious features. {sections 5.2.3.7, 5.2.3.8, 5.2.3.9}.
- Recognition was not destroyed despite 'physical noise' in the form of swarf strewn over the object. {section 5.2.3.9}.

Sensitivity to internal parameters

- The system operated without change when 3 other edge detectors were substituted for the Walsh transform based edge detector. {section 5.2.4.1}.
- The system showed some sensitivity to the size of the local neighbourhood radius. {section 5.2.4.2}.

Tests on the edge detector

- Details of the performance of the Walsh edge detector and the

other edge detectors were given in section {5.3.1}. The performance appears to be satisfactory.

Tests on the rep-point algorithm

- Details of the performance of the rep-point algorithm were given in section {5.3.2}, which shows that rep-points are more reliable, more repeatable and less noisy than edge points.

Tests on the learning algorithm

The learning principle of the system was tested in many ways:

- The cutter (CUTTER), and the cutter at low light intensity (CUTLL) were taught as separate objects. This library was then used to recognize the cutter when the lighting was changed. The results were as expected; CUTTER was recognized when the light intensity was high, followed by a gradual transition to CUTLL when the light intensity was reduced. {section 5.2.1.1}.
- The same experiment was repeated by teaching the cutter and a low scale version of the cutter as separate objects. {section 5.2.1.2}.
- The cutter and the tooth wheel were taught on a chess board. The system was able to reject the chess board features as they were common between the two sets of images. {section 5.2.3.5}.
- The system was able to learn and recognize a handful of swarf on a grey metallic background as an 'object'. {section 5.2.3.9}.

Execution speed

- Section {5.4} describes the execution speed tests.

Chapter 6

.uture Work and Conclusions

This chapter is organized as follows: section {6.1} looks at future work in terms of architectural extensions. Section {6.2} gives a hardware design for a pre-processor. Section {6.3} concludes this thesis.

6.1. Future work: Extending the architecture

The architecture described so far was that of the implemented system. However, as pointed out in section {3.6.5}, this architecture has four main limitations. In this chapter I propose ways of removing some of these limitations,

(a) by exploiting further the principles described in chapter 2, and

(b) by using the flexibility achieved by the implemented system.

The four main limitations are:

(a) The restriction to unique local structure, and therefore the inability to respond to objects with only unique global features.

(b) Restriction to a constant scale factor.

(c) Limitation to stable states of objects.

(d) The inability to cluster objects into classes.

6.1.1. Extending the system to cope with 'simple' objects

An unusual limitation of the present system is that it is not able to recognize objects with simple shapes. For example, the system cannot recognize a (large) square from a (large) rectangle because there are no unique local neighbourhoods. This is because the present implementation and architecture are limited to the use of local subgraphs. It is clearly important to use the global relationships between local neighbourhoods.

One possibility is to enhance the present system using concurve descriptions. Such descriptions have been well tested and are known to work very well for simple shapes {section 3.7.1}. The local feature description of the present system would complement the concurve description to form a powerful new vision system.

However, I propose that the original principles developed in chapter 2 be used, and that the system be extended to find unique non-local subgraphs. As described in section {2.1.2} the main problem in doing so is the explosion in the number of subgraphs that have to be tested for in the learning stage. Therefore it is proposed that the combinatorics be controlled by using arbitrary rules to limit the number of subgraphs that have to be considered.

This may be done by introducing the notion of intermediate features. An intermediate feature is formed using local features in the same way as local features are formed from rep-points. (See Fig. 6-1). The radius of an intermediate feature is, of course, larger than that of a local feature. Now, intermediate features can be treated exactly the same way as local features, using the same match-

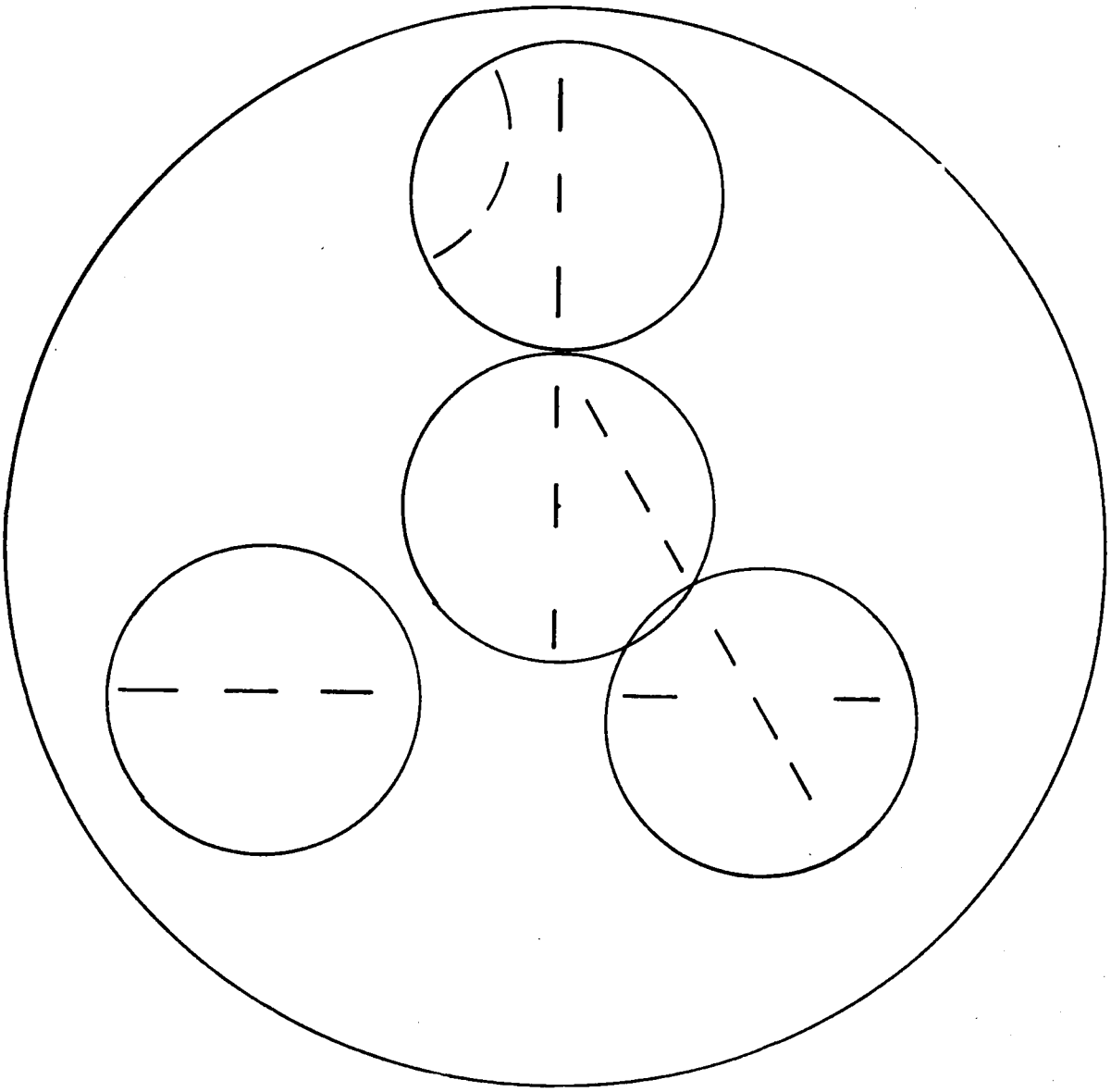


Fig. 6-1 An intermediate feature

ing algorithm except that instead of matching rep-points, local features will be matched. Thus the matching algorithm will be called recursively. Note that the combinatorial explosion of the graph matching problem is controlled by the severe restriction on local features that can be matched, and by the rapid tests that can be used to check for intermediate features that do not match i.e. intermediate features match only if,

1. the two intermediate features have the same number of local features, and
2. the two central local features are matched (this is an extra condition that is not present for local feature matches, as all rep-points match each other once),
3. the peripheral local features cover each other i.e. the peripheral local features are matched, and they are approximately the same orientation and position relative to the central local feature.

Intermediate features may be chosen whenever local features are chosen so that the system would have an equal number of intermediate features, local features, and rep-points. This method of choosing intermediate features has the same advantages as for choosing local features {section 3.2.4}. If this leads to too many intermediate features being chosen for the available resources, intermediate features may be chosen around unique or uncommon local features only. The learning algorithm can be used to find unique intermediate features in exactly the same way as for unique local features. The number of unique intermediate features found is expected to be more than the number of unique local features because the intermediate feature 'sees' more of the object. Therefore the motivation behind the use of intermediate features is to allow the features to see larger neighbourhoods. In that case, is it not possible to simply increase the radius of the local neighbourhoods instead? This is indeed a possibility, especially as the execution time does not increase dramatically with radius. This is due to the restrictions on neighbourhoods that need to be compared due to variations in the number of rep-points in a local neighbourhood. (However, the time

required to compare two local neighbourhoods that eventually match increases with the number of rep-points in the neighbourhood.) The disadvantage with using larger local neighbourhoods is that they will no longer be 'local'. Therefore they will become more sensitive to object obscuration etc., so that the probability of a local neighbourhood being affected by visual disturbances is increased. I would therefore like to keep the local neighbourhoods at approximately the present size. An alternative would be to use several local neighbourhood sizes simultaneously. That is the same as constructing the intermediate features with rep-points rather than with local features. The difference is not dramatic, but I favour the first approach of using local features to form intermediate features, in order to reduce the significance of the central rep-point on intermediate feature orientation and position, and to reduce the significance of any particular rep-point on the intermediate feature match.

Clearly then, further levels of features may be formed by using intermediate features to create larger features. I feel, however, that two levels of local features are sufficient, but I would like to introduce the idea of global features. These are features formed from local features (or intermediate features) that are far apart on the object. I propose that arbitrary rules should be used to determine the number of local features within a global feature and the method of choosing them, as in Stockman et al {section 3.7.2}. If processing resources at learning time allow it, all combinations of 3 uncommon local features can be used. The reason for using global features, rather than a relational search of local features at recognition time, is to allow the learning algorithm to identify unique global

structure. Therefore the difference between a square and a rectangle would be detected using unique global features. It is felt that the graph matching problem at learning time can be effectively controlled in this way (cf. general proposal in section {2.1}).

6.1.2. Coping with Scale Variations

When designing for scale variations, it is necessary to talk of the extent of scale variation that needs to be handled, as it is clearly not possible to cope with the full range of scale variations. How then can a vision system be designed to handle as large a scale variation as possible? The vision system is already able to cope with small variations of scale of up to about 30% {section 5.2.1.2}. One way of extending this is to teach each object at different scales as examples of the same object. Reliability tests will not be done across examples so that new unique features will be formed at different scales. (However, the reliability test should be done within each example to verify that a ±30% scale variation can be tolerated by all the chosen features.) Therefore, examples can be used to tailor the match response of the system. Although the present implementation does not allow this to be done explicitly, I have tested this possibility by showing images of the object 43% smaller as non-examples. This reduces the match flexibility as expected {section 5.2.1.2}. Therefore, objects can be taught at scale intervals of 30%-50% (so that each model has to cope with a scale variation of ±15% to ±25%). The usable scale of a 256x256 image could be covered by about 5 models per object.

6.1.3. Coping with 3D orientation variation

The vision system is at present limited to the recognition of objects in their stable states or just outside their stable states (by up to about 30° {section 5.2.1.3}). How can the system be extended to cope with 3D orientation variation over the possible range? First an important point about 3D views of objects: Researchers have for many years been used to the idea of representing 2D views of objects by a finite set of pixels (say 128x128) i.e. the approximation was acceptable. Clearly then, the 3D viewing angle could be quantized too, so that a tolerable approximation of an object can be formed from multiple 2D views. The proposal is to represent objects by a relational structure of local features over the surface of the object. How many views of the object do we need? Assuming that local feature descriptions can be chosen to be invariant through object rotations of from 20°-30° only a small number of views are necessary. (Note that the learning algorithm has to be extended so that it verifies that the chosen features are structural through the assumed angular variation.)

There are two constraints placed on the number of 3D views needed.

1. The area covered by any one view must not be more than that due to a rotation of the object by more than 20°-30° from the centre of the view.
2. The number of views must be manageable.

Using the lower limit of 20° allowed for the rotation, I now try to compute (approximately) the number of views needed.

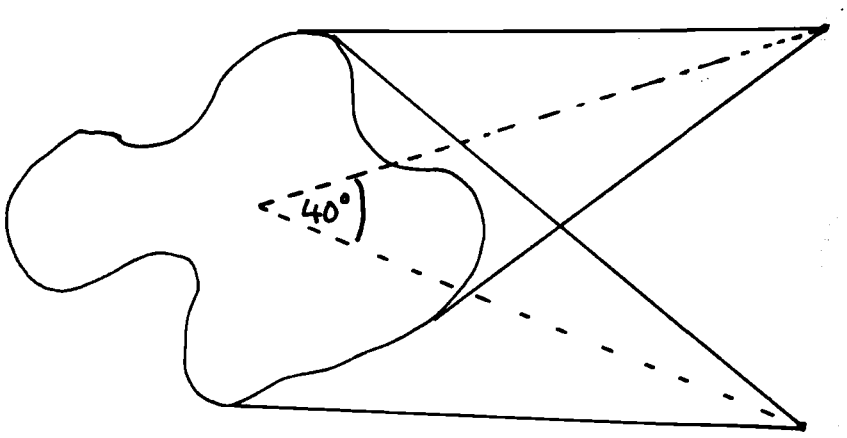


Fig. 6-2

Constraint 1 requires that the angle between two views be not greater than 40° (see Fig. 6-2) (i.e. each view copes with an object rotation of ±20° in any direction). Therefore the question is, how many vectors can we draw emanating from the centre of the object such that no two vectors are less than 40° apart? The problem may be worded differently to allow an approximate answer to be computed easily. How many squares can be drawn on a sphere, so that the angle subtended at the centre of the sphere by the sides of the square is 40°?

area of square = $2R.\tan(20°)^2$

area of sphere = $4\pi R^2$

hence number of squares = $\dfrac{\pi}{\tan(20°)^2}$

= 23.7

Therefore it is possible to cover an object with about 24 views (10 views if a figure of 30° is chosen) such that the neighbouring squares of any square is less than 20° away (see Fig. 6-4).

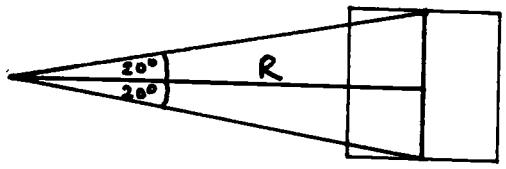


Fig. 6-3

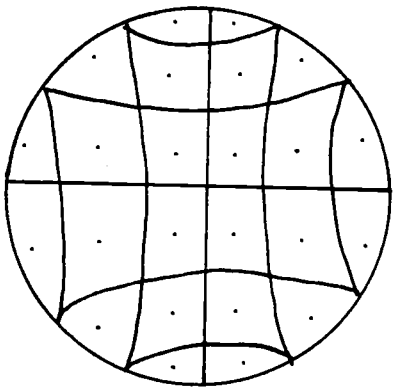


Fig. 6-4

In the object learning stage the 24 views of the object can be taught to the system. Once an object is represented by a relational structure (Fig. 6-5) of local features, the learning algorithm should find unique local features, and unique intermediate features. If unique global features are used, it is necessary to ensure that the global features are visible from a single view.

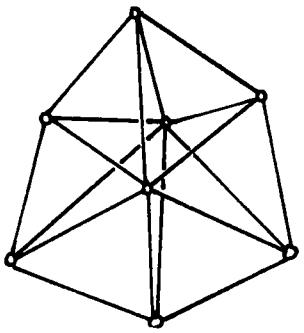


Fig. 6-5

6.1.4. The Need for Clustering Objects

If the present system is given the task of recognizing 5 different objects, with two of the objects being very similar to each other so that (say) only 2 unique features are found for these two objects, the system would not be able to recognize these two objects (under poor operating conditions) even if these two objects, as a class, were very different from the rest of the objects. One solution is to place the two objects (A and B) in a single class, and then find unique structure to separate class AB from the other objects. A different set of features (including the two unique features found earlier) could then be used to separate A from B once the class is recognized.

The feature sets U2 {section 4.3.2} were generated for this reason. Each set in U2 describes the similarity (and difference) between two of the learned objects. These sets could be used to cluster objects that are similar. Each class would then be treated as a new

composite object. A new set of unique features will be generated within each class to separate the objects in the class.

6.1.5. Possible Application to Scene Analysis and 'Very High Level' Vision

I feel that local and intermediate features may be of use in producing hypothesis of what may be in the scene for general scene analysis work. They can be used to choose frames† rapidly. The main advantage is the rapid execution speed possible, especially on parallel processors, so that a large number of unique local features from completely different contexts can be tested for at high speed. Any matches found can be used to inject asynchronous hypothesis to a scene analysis program.

† A frame is a data structure for representing information about a particular situation. See Minsky [1975].

6.2. Future Work: Design of a Hardware Pre-Processor

In this section I propose a hardware implementation for the algorithms described in this thesis. This proposal is based on experience gained by the author during the design and implementation of a hardware vision processor which is described in detail in Athukorala [1981].

6.2.1. Implementation using a Cellular Array Processor

The vision algorithms described in chapter 3 are well suited for implementation on a cellular array processor.† The data should be organized so that each processor is responsible for a single local neighbourhood. The complete system including the recognition algorithm could be implemented on such a processor. For example, it would be possible to give the list of unique local features to each processor in the array, so that each processor can search for this list independently in its section of the image. The pre-processing could also be carried out by the same processor. Clearly, such a strategy would make good use of the resources of the array processor. Thus, very high resolutions could be handled by increasing the number of processing elements used, without a significant increase in execution time.

6.2.2. Pipelined Implementation

Despite these advantages with array architectures, a pipelined architecture is preferred at present for three reasons.

---

† See Duff [1982], Hunt [1981] and Potter [1982] for examples of array processors.

1. The ability to use off-the-shelf components.
2. Lower expected cost.
3. Flexibility of implementation (i.e. the problem could be tackled in stages using a modular approach).

The proposed implementation is based on the following basic architecture. (Fig. 6-6). The complete pre-processor will be based on several of these blocks used in a pipeline. The data buffers are used to link processing stages of different speed, and to provide parallel access to data.

As we have seen {section 3.2}, the pre-processor consists of the following stages.

1. Walsh transform edge detection.
2. Rep-point selection.
3. Neighbourhood selection and normalization.

In the following, I will assume that the processing is to be done at video speed (20ms/frame) on 256x256 images. The pixel time will be approximately 200ns. I will also assume that the processing will be carried out using high speed 32 bit microprocessors (such as the Motorola 68020) whenever possible. It will be noticed that special

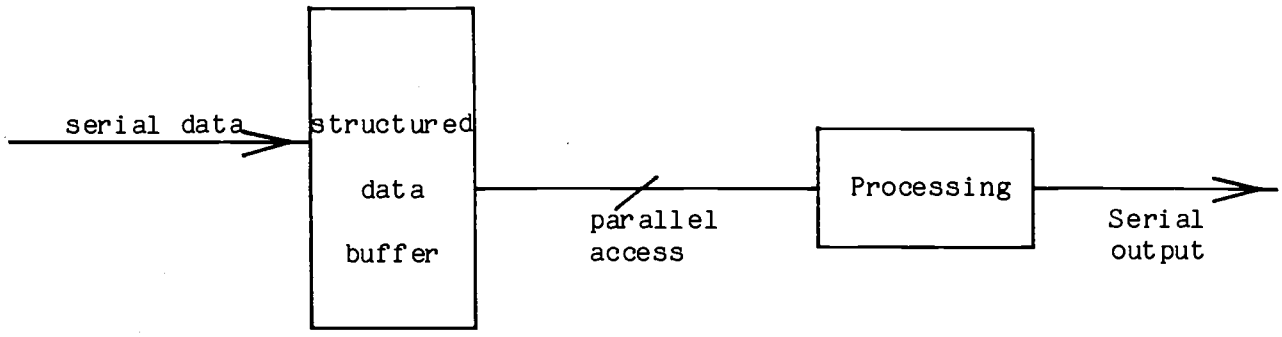


Fig. 6-6 Architecture of a single stage of the pipeline

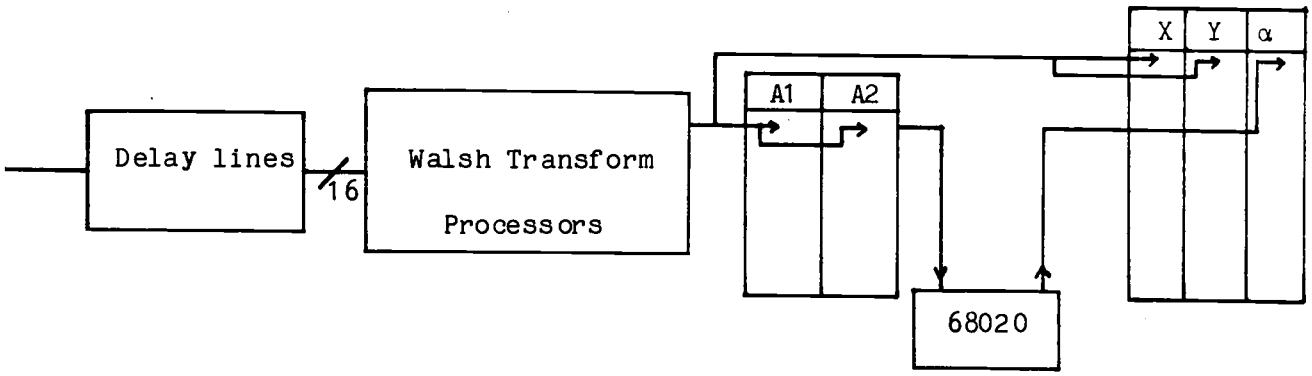
purpose hardware will be needed at the beginning of the pre-processor due to the large processing requirement, while later processing could be carried out using one or more microprocessors.

6.2.2.1. Walsh Edge Detection

This algorithm requires a very high processing rate, which can be provided (at present) only by using special purpose hardware. The input data buffer will be a delay line structure providing a 4x4 window of pixels. The following processing is needed for each pixel.

1. 43, 16 bit additions (and subtractions).
2. 5 absolute value computations.
3. 1 division.
4. 2 comparisons.

At this stage, an edge point is found. The following processing is



A1,A2 - First two Walsh coefficients
X,Y - Address of edge points
α - Edge orientation

Fig. 6-7 The edge detection stage

required on each edge point.

5. 1 division.

6. An inverse tangent operation.

7. Quantization to 8 bits.

Processes 1-4 above will have to be implemented using special purpose hardware. The edge detection stage could be simplified by using some other edge detector (e.g. Sobel), but would result in an increase of the processing requirements placed on the rep-point stage (as these edge detectors result in much thicker -and therefore numerous- edges), which would be an unwise choice, as the Walsh edge detector is simpler to implement than the rep-point algorithm. Therefore an increase of processing at the rep-point stage may require it to be implemented in special purpose hardware as the microprocessors may no longer be able to cope with the processing demand. Given this choice, it would be simpler to implement the Walsh edge detector in special purpose hardware, than to implement the rep-point algorithm in special purpose hardware.

Processes 5-7 could be implemented with a single microprocessor using a look up table to achieve processes 6 and 7. This is possible due to the relatively small number of data points (~4000) that have to be processed. Process 5 may need special arithmetic support. A 68020 operating at 20MHz would be able to compute the angle data for about 4000 edge points in the 20ms frame time. (It should be noted that this is possible due to the limited resolution -8 bits- of the computations, which allows the use of small look up tables etc.). Fig. 6-7 is a block diagram of the edge detection stage. The A1A2 buffer and the y- uffer are FIFO buffers accessed via pointers. The pointer

manipulations (i.e. incrementing, decrementing and clearing) could be accomplished by the hardware, so that the 68020 need not be aware of their presence. The 68020 will simply perform memory reads and writes to and from reserved locations. The resulting edge data will be stored in the XYα data buffer which will be used by the rep-point stage.

6.2.2.2. The Rep-Point Algorithm

The rep-point algorithm may be implemented using two processors in a manner similar to the software implementation. The first processor will be used to segment the rep-points in the horizontal direction, and the second processor will collect vertically related 1D rep-points to form 2D rep-points. Fig. 6-8 is a block diagram of the rep-point stage.

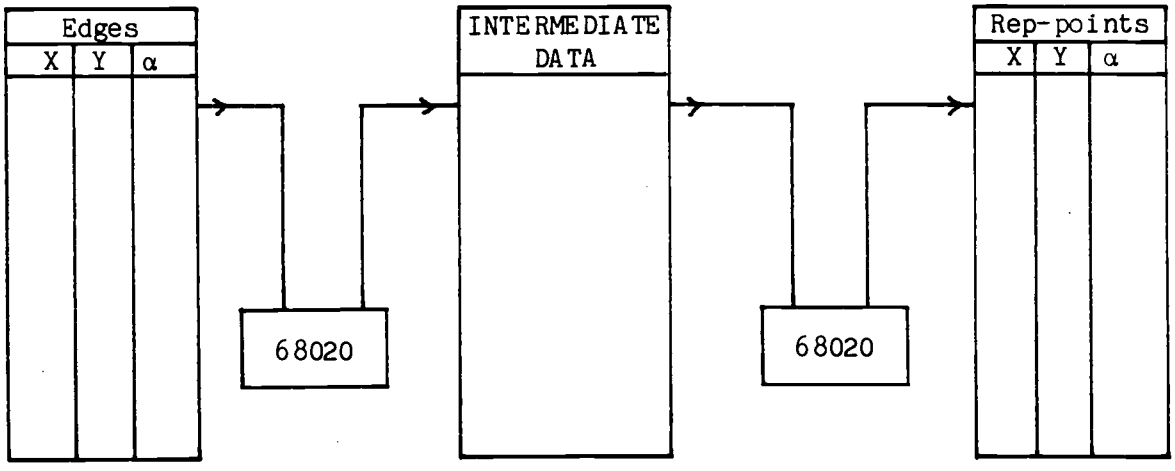


Fig. 6-8 The rep-point stage

6.2.2.3. The Local Neighbourhood Algorithm

This algorithm can be implemented in the same way as the rep-point algorithm. However, only a single 68020 will be needed as only a small number of data points are to be processed (usually about 200-400).

6.2.2.4. Overall System Implementation

Fig. 6-9 shows the overall system implementation. The final 68020 will be responsible for executing the recognition algorithm and the learning algorithm. It will also perform the system control functions and communicate with the outside world.

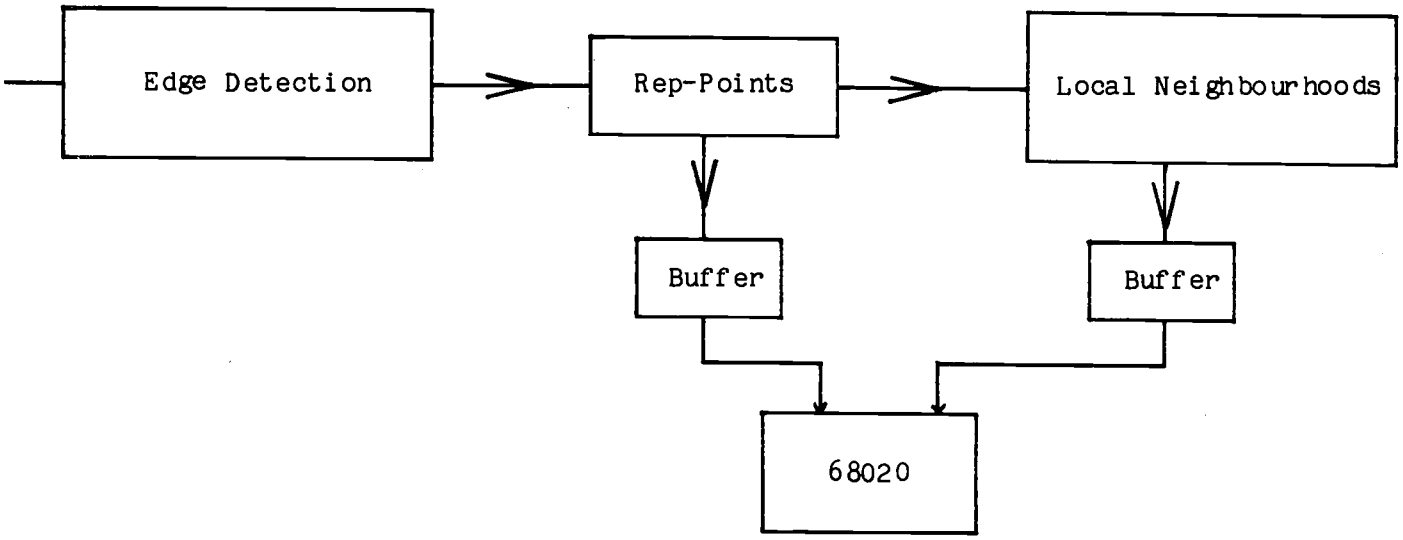


Fig. 6-9 Block diagram of overall system implementation

6.2.3. Discussion of the Architecture

An important feature of this architecture is its flexibility for implementation. For example, a basic system could be implemented using a single 68020 plus the front-end of the Walsh processors. This would result in a much slower, but cheaper implementation. Each stage of the pipeline could then be added, so that the burden on the final 68020 is gradually reduced. The full configuration would be able to operate at video speed. However, it should be noted that this depends on the complexity of the input image. The system will be able to operate at video rates provided that there are less than a pre-specified number of edge points in the image. If the number of edge points exceeds this threshold, the system could be designed to take one of two courses of action. It could ignore the lower part of the image that has not been processed, or ignore the next frame of data. If the first course is taken, the system would be designed so that the complete pre-processor is reset after each frame, so that unprocessed edge data is discarded. Alternatively, if the next frame of data is discarded, the extra time could be used to process the old frame. This could be extended to as many frames as necessary. This facility could be program selectable.

Another advantage with this configuration is that the system could be easily extended to cope with larger image resolutions, as only the delay line structure needs to be changed. (The Walsh transform processors may need to be replaced as well if there is a substantial change in the pixel frequency).

Finally, the system may be implemented using custom VLSI chips, or special hardware instead of the microprocessors. This implementa-

tion too could be achieved in stages beginning from the front-end of the processor. This then points to a clear upgrade path for product development. The penalty paid for this flexibility is the need for large amounts of buffer memory. However, this may not be a cause for concern if the present trend of falling memory costs continues.

6.3. Conclusions

6.3.1. Contributions

This thesis was concerned with the problem of recognizing industrial objects rapidly and flexibly. These objectives were achieved using a general strategy based on a generalized local feature detector, an extended learning algorithm, and the use of unique object structure. The main contribution of this thesis is the overall strategy that allowed flexible and fast operation of the system; It was shown in section {3.7} that the system performance compares favourably with previously reported vision systems.

The task of the generalized local feature detector is to generate a highly descriptive representation of local object structure so that the description is independent of the imaging conditions of interest. (Object structure was defined as everything about the object that is visible and independent of the imaging conditions of interest {2.1.1.1}). This is achieved by first using an edge detection operation to reduce the sensitivity to absolute lighting level, and to achieve a degree of data reduction with a minimum loss of useful information. A new algorithm called a rep-point algorithm is then used to find representative points for small areas of approximately uniform edge property. These rep-points form the elementary features of the system. In the next stage, the rep-points are used to form local features by using each rep-point as a focal point for choosing a local neighbourhood. Thus, local features are local subgraphs of the rep-point relational structure. Locality is defined by the spatial distance between the rep-points. This method of feature detection was

seen to have many advantages:

1. The feature descriptor is able to represent complex object structure. No restrictions are placed on object complexity except due to image resolution i.e. the feature detector is not limited to the detection of straight lines, circular arcs, and 'conventional' local features, and therefore is able to operate on objects that do not have a significant amount of these features {5.2.3.9}.
2. The features are insensitive to a variety of imaging conditions including feature position and orientation in 2D, and overall lighting level {3.2.1}.
3. The pre-processor algorithms are inherently parallel as all computations are based on local neighbourhoods. This makes the pre-processor ideal for implementation on special hardware architectures.
4. The feature descriptor, along with the feature matching algorithm {3.3}, form a new generalized feature detector that may be of general interest.

The performance of the overall system is attributed to the extended learning strategy. The learning strategy is based on a reliability test and on finding unique structure of the learned objects. The task of the reliability algorithm is to observe the performance of the pre-processor over the imaging conditions of interest, and select a set of features that are reliably reproduced by the pre-processor. In this way the system compensates for pre-processor imperfections, and improves the recognition reliability.

Unique structure of an object is a set of descriptions of subparts of the object that remain unique over the imaging conditions of

interest. Unique structure is found as follows: Once all the object instances have been represented by a relational structure of elementary features, all possible subgraphs are formed. Reliable subgraphs are then searched for in the subgraphs found for the other objects in the library. All subgraphs that do not find a match are unique to the original object. In order to reduce the combinatorics of this strategy, subgraphs are limited to local features generated by the preprocessor.

This strategy has many advantages:

1. The graph matching problem is transferred from the recognition stage to the learning stage, as the recognition algorithm has to search only for unique subgraphs {3.4.2}.
2. It is sufficient to find just one unique subgraph for recognition, which allows rapid execution of the recognition algorithm. However, since it is not possible to guarantee a perfect reliability test, more than one unique feature is required to confirm recognition when ideal operating conditions cannot be guaranteed.
3. This also allows the system to operate flexibly, as it is able to reach a 100% confidence level of recognition even if a substantial number of unique features are lost due to object obscuration or degraded operating conditions.
4. The reliability test allows the system to compensate for preprocessor imperfections. It also allows the system to extract the common features from an object that may itself be variable from one instance to another (perhaps because of manufacturing variations). Further, the reliability test allows the system to discard background features during the learning stage {3.4.2}.

5. The user is given advanced warning of the system performance on the particular object library that was chosen {3.4.2}.

The task of the recognition algorithm was to search for unique features, and execute rapidly. It was shown that the execution speed could be minimized by the use of various heuristics to respond to particular situations. When high contrast images were guaranteed, the system was able to execute rapidly (10ms for a small object {section 5.4.3}), while the time taken to search for all unique features of 3 objects was 1-5s. Therefore, it was shown that the system could be easily configured to operate rapidly with high quality images, or flexibly in poor conditions {4.4}.

The test data shows that the system displays a significant degree of insensitivity to variations in its three main assumptions {5.2.1}: constant lighting (up to 70% reduction), constant scale (up to 30% reduction), and 2D views (30°-40° outside the learned 2D plane). The system was also able to demonstrate a degree of insensitivity to a variety of other operating conditions {5.2} such as the addition of Gaussian noise (signal to noise ratio as low as 8.5dB). This performance demonstrates that the generalizations made by the learning algorithm hold, not only within the domain of the sampled images, but well outside this domain. Thus the system demonstrates a real learning capability.

In order to remove some of the limitations of the system, it is necessary to extend the learning algorithm further, so that

1. the reliability test is carried out over a larger range of variations in imaging conditions,

2. unique intermediate structure and unique global structure is used to allow objects without unique local structure to be recognized {6.1.1},
3. scale and 3D orientation independent recognition is obtained by teaching several views of the same object at different scales and 3D orientations {6.1.2, 6.1.3},
4. similar objects are clustered into classes so that new composite objects may be formed from these classes {6.1.4}

Thus, the overall strategy of the system {2.1} is to learn automatically what makes an object unique over the expected variation of imaging conditions. This is achieved by exhaustive subgraph isomorphism in the learning stage to find reliable and unique subgraphs. It is believed that this technique is of general interest when the objects to be recognized can be represented by a set of features, a subset of which describe the objects sufficiently well, and are independent of the imaging conditions through the required range. The particular feature detector used is also thought to be of general interest for vision work as it allows a much larger range of local features to be used than before {2.1.1}.

References

Abdou J.E. and Pratt W.K. [1979], Quantitative design and evaluation of enhancement/ thresholding edge detectors, Proc. of the IEEE, Vol 67, No. 5, May 1979, pp. 753-763

Agin G.J. [1975], 'An experimental vision system for industrial application', 5th Int. symp. on Industrial Robots, Sept. 1975, pp. 135-148

Agin G.J. and Binford T.O. [1973], 'Computer recognition of curved objects', Proc. 3rd int. jnt. conf. on artificial intelligence, Stanford, CA, pp. 641-647.

Ambler A.P, Barrow H.G, Brown C.M, Burstall R.M, and Popplestone R.J. [1975], 'A versatile system for computer controlled assembly', Artificial Intelligence 6, pp. 129-156

Antonsson D., Danielsson P-E, Gudmundsson B., Hedblom T., Kruse B., Linge A., Lord P., and Ohlsson T. [1981], 'PICAP - A system approach to image processing', Proc. computer architecture for pattern analysis and image data base management, Virginia, USA, IEEE, pp. 35-42

Athukorala A.S. [1980], 'Low level vision', DAI Working paper no: 66, Dept. of Artificial Intelligence, Univ. of Edinburgh, UK.

Athukorala A.S. [1981], 'Some hardware for computer vision', DAI working paper no: 102, Dept. of Artififcial Intelligence, Univ. of Edinburgh, UK.

Athukorala A.S. [1985], 'An essay on the aspects of the human visual system that influenced my thesis work', DAI working paper, Dept. of Artificial Intelligence, Univ. of Edinburgh, UK.

Athukorala A.S. and Wallace A.M. [1982], 'Image processing for industrial component identification using hardware techniques', Int. conf. on electronic image processing, pp. 129-133, July 1982.

Baird M.L. [1982], 'GAGESIGHT: A computer vision system for automatic inspection of instrument gages', IEEE conf. on industrial applications of machine vision, pp. 108-111

Ballard D.H. [1979], 'Generalizing the Hough transform to detect arbitrary shapes', TR-55, Computer Science Dept., Univ. of Rochester, USA, Oct. 1979

Barnard S.T. [1980], 'Automated inspection using gray-scale statistics', Proc. 1st annual national conf. on Artificial Intelligence, Stanford Univ., pp. 49-52

Beattie R.J. [1984], 'Edge detection for semantically based early visual processing', PhD thesis, Dept. of Artificial Intelligence, Univ. of Edinburgh.

Bolles R.C. [1979], 'Symmetry analysis of 2-dimensional patterns for computer vision', Tech note:186, SRI International, June 1979.

Bolles R.C. and Cain R.A. [1983], 'Recognizing and locating partially visible objects: The local-feature-focus method', in Robot vision, ed. A Pugh, Springer-Verlag.

Brauner R. [1982], 'Automated chip (die) inspection', IEEE conf. on industrial applications of machine vision, Research triangle park, N.C., USA, May 1982, pp. 43-50

Bron C. and Kerbosch J. [1971], 'Algorithm 457: Finding all cliques of an undirected graph [H]', Communications of the ACM, Vol 16, No. 9, Sept 1973, pp. 575-597

Brooks R.A., Greiner R., Binford T.O. [1979], 'The ACRONYM model-based vision system', 6th Int. jnt. conf. on AI, Tokyo, Japan, pp. 105-113

Cheng J.K. and Huang T.S. [1981], 'Image recognition by matching relational structures', Proc. IEEE conf. on pattern recognition and image processing, Dallas TX, Aug 1981, pp. 542-547

Cheng J.K. and Huang T.S. [1982] 'Image registration by matching relational structures', 6th Int. conf. on pattern recognition, Munich 1982, IEEE, pp. 354-356

Chin R.T. and Harlow C.A. [1982], 'Automated visual inspection: A survey', IEEE trans. on pattern analysis and machine intelligence, Vol PAMI-4, No 6, Nov 1982, pp. 557-573

Cohen P.R. and Feigenbaum E.A. [1982], The handbook of AI, Vol. 3, Pitman 1982.

Corneil G. and Gotlieb C.C. [1970], 'An efficient algorithm for graph isomorphism', Jnl. of the ACM, Vol. 17, No. 1, Jan 1970, pp. 51-64.

Cornsweet T.N. [1970], Visual perception, Academic press

Davis L.S. [1975], 'A survey of edge detection techniques', Computer graphics and image processing, Vol. 4, pp. 248-270.

Dessimoz J-D, Kunt M., Zurcher J.M., Granlund G.H. [1979], 'Recognition and handling of overlapping industrial parts', 9th Int. symp. on industrial robots, pp. 357-366

Devijver P.A. and Kittler J. [1982], Pattern recognition: A statistical approach, Prentice-Hall.

Duff M.J.B. [1982], 'Special hardware for pattern processing', 6th Int. conf. on pattern recognition, Munich, IEEE, pp. 368-379

Freeman H. [1961], 'On the encoding of arbitrary geometric configurations', IRE trans. elec. comp., EC-10, pp.260-268, June 1961

Fu K.S. [1982], Syntactic pattern recognition and applications, Prentice-Hall.

Giralt G., Gallab M., and Stuck F. [1979], 'Object identification and sorting with an optimal sequential pattern recognition method', 9th int. symp. on industrial robots, pp. 379-389.

Hara Y., Akiyama N., Karasaki K. [1982], 'Automatic inspection system for printed circuit boards', IEEE conf. on industrial applications of machine vision, pp. 62-70.

Hattich W. [1982], 'Recognition of overlapping workpieces by model directed construction of object contours', Digital systems for industrial automation, Vol. 1, Part 2, Nos. 2.3, pp. 223-239.

Hilditch C.J. [1969], 'Linear skeletons from square cupboards', Machine Intelligence 4, eds. Meltzer and Michie, pp. 403-420.

Horn B.K.P. [1974], 'Determining lightness from an image', Computer graphics and image processing, Vol 3(1), pp. 277-299.

Hough P.V.C. [1962], 'Method and means for recognizing complex patterns', US patent 3069654, 18 dec 1962.

Hueckel M. [1973], 'An operator which locates edges in digitized pictures', Jnl. of the ACM, Vol 18 (1), Jan 1971, pp. 113-125.

Hunt D.J. [1981], 'The ICL DAP and its application to image processing', in Languages and architectures for image processing, eds. Duff M.J.B. and Levialdi S., Chapter 22, Academic press

Igarachi K, Naruse M, Miyazaki S and Yamada T [1979], 'Fully automated integrated circuit wire bonding system', 9th int. symposium on industrial robots, pp. 87-97.

Jacobus C.J. [1979], 'Visual recognition of artifacts by computer', PhD thesis, Univ. of Illinois at Urbana-Champaign, USA.

Jacobus C.J. and Chien R.T. [1979], 'Recognition of visual forms by features drawn from graphical relations', Proc. of the int. conf. on cybernetics and society, IEEE, Denver CO., USA, Oct 1979, pp. 50-55

Karg R. and Lanz O.E. [1979], 'Experimental results with a versatile optoelectronic sensor in industrial applications', 9th int. symposium on industrial robots, pp. 247-264

Kelley R, Birk J, Duncan D, Martins H and Tella R [1979], 'A robot system which feeds workpieces directly from bins into machines', 9th int. symposium on industrial robots, pp. 339-355

Kimura F, Yoshimura M and Miyake Y [1982], 'An algorithm for subpattern matching of line patterns', 6th int. conf. on pattern recognition, IEEE, Munich, pp. 461-464.

Knuth D. [1969] The art of computer programming, Vol 2, Addison-Wesley.

Konishi T, Misono M and Kato T [1982], 'A new technique for inspecting CCD wafers for defects', IEEE conf. on industrial applications of machine vision, pp. 51-54

Kopolowitz J. [1981], 'On the performance of chain codes for quantization of line drawings', IEEE trans. on pattern analysis and machine intelligence, PAMI-3, No 2, March 1981, pp. 180-185.

Kruger R.P. and Thompson W.B. [1981], 'A technical and economic assessment of computer vision for industrial inspection and robotic assembly', Proc. of the IEEE, Vol 69, No 12, December 1981, p. 1524-1538.

Malinen P. and Niemi A. [1979], 'Reduction of visual data by a program controlled interface for computerized manipulation', 9th int. conf. on industrial robots, pp. 391-403.

Marr D. [1974], 'An essay on the primate retina', MIT AI Lab, Memo No. 296, Jan 1974

Marr D. and Hildreth E. [1979], 'Theory of edge detection', MIT AI Lab, AI memo no. 518, April 1979

Mckee J. and Aggarwal R. [1977], 'Computer recognition of partial views of curved objects', IEEE trans. on computers, Vol C-26, No 8, pp. 790-800

Mero L. [1981a], 'An optimal line following algorithm', IEEE trans. on pattern analysis and machine intelligence, Vol PAMI-3, No 5, Sept 1981, pp. 593-598.

Mero L. [1981b] 'An algorithm for scale - and rotation - invariant recognition of two dimensional objects', Computer graphics and image processing, Vol 15, pp. 279-287

Minsky M. [1975], 'A framework for representing knowledge', in The psychology of computer vision, ed. Winston P.H., McGraw hill.

O'Gorman F. [1978], 'Edge detection using Walsh functions', Artificial Intelligence, Vol 10, pp. 215-223

Olsztyn J.T, Rossol L, Dewar R, Lewis N.R [1973], 'An application of computer vision to a simulated assembly task', Proc. int. joint conf. on pattern recognition, Washington DC, USA, pp. 505-513

Olympief S, Pineda J.C, Horaud P [?], 'The use of vision for industrial inspection', Lab. d'Automatique de Grenoble, ENSIEG, BP 46, 38402 St Martin d Hères, France. Date not known.

Osteen R.E. and Tou J.T. [1973], 'A clique detection algorithm based on neighbourhoods in graphs', Int. jnl of computer and information sciences, Vol 2, No 4, pp. 257-268.

PAMI [1983], Pattern analysis and machine intelligence, Vol PAMI-5, No 6, Nov 1983, pp.561-630.

Pavlidis T. [1978], 'Survey: A review of algorithms for shape analysis', Computer graphics and image processing, Vol 7, pp. 248-258.

Pavlidis T. [1980], 'Algorithms for shape analysis of contours and waveforms', IEEE trans. on pattern analysis and machine intelligence, Vol PAMI-2, No 4, July 1980, pp. 301-312.

Perkins W.A. [1977], 'Model based vision system for scenes containing multiple parts', 5th int. joint conf. on AI, Cambridge, Mas, USA. pp. 678-684.

Perkins W.A. [1978] 'A model based vision system for industrial parts', IEEE trans. on computers, Vol C-27, No 2, Feb 1978, pp. 126-143

Perkins W.A. [1983], 'INSPECTOR: A computer vision system that learns to inspect parts', in IEEE trans. on pattern analysis and machine intelligence, Vol PAMI-5, No 6, pp. 584-592.

Persoon E.H.J. [1978/9], 'A system that can learn to recognize 2-dimensional shapes', Philips tech. review, 38, No. 11/12, pp. 356-363

Potter J.L. [1981], 'Continuous image processing on the MPP', Proc. computer architecture for pattern analysis and image data base management, Virginia, USA, IEEE, pp. 51-56

Potter J.L. [1982], 'Pattern processing on STARAN', in Special computer architectures for pattern processing, eds. Fu K.S. and Ichikawa T., Chapter 5, CRC press

Pratt W.K. [1978], Digital image processing, John Wiley and sons.

Presern S. and Kandus G. [1981], 'Object recognition by computer vision. An application of intelligent measurement system in industry', 1st IMEKO summer school on the application of microcomputers in measurement, Dubrovnik, Yugoslavia, Sept 1981. pp. 71-75

Raggett D. [1980], 'A survey of computer vision research', Machine Intelligence Research Unit, Edinburgh University, Sept 1980.

Rummel P. and Beutel W. [1982], 'A model based image analysis system for workpiece recognition', 6th int. conf. on pattern recognition, IEEE, pp. 1014-1017

Searle N.H. [1969], 'Shape analysis by use of Walsh functions', Machine Intelligence, Vol 5, pp. 395-410

Shirai Y. [1973], 'A context sensitive line finder for recognition of polyhedra', Artificial Intelligence, Vol 3, pp. 95-119.

Shirai Y. [1978], 'Recognition of real-world objects using edge cues', in Computer vision systems, eds. Hanson A.R. and Riseman E.M., Academic press

Sobel I. [1970], 'Camera models and machine perception', Stanford AI project, AIM-121, Dept. of Comp. Science, Stanford univ., Stanford, CA, USA.

Stockman G, Kopstein S, and Benett S. [1982], 'Matching images to models for registration and object detection via clustering', IEEE trans. on pattern analysis and machine intelligence, Vol PAMI-4, No 3, Jan 1982, pp. 229-249.

Taylor W.K. and Ero G. [1980], 'Real time teaching and recognition system for robot vision', The industrial robot, June 1980.

Tropf H. [1981], 'Analysis-by-synthesis search to interpret degraded image data', 1st int. conf. on ROVISEC, Stratford-upon-Avon, UK, April 1981, pp. 25-33.

Ullman J.R. [1976], 'An algorithm for subgraph isomorphism', Jnl. of the ACM, Vol 23, No 1, Jan 1976, pp. 31-42.

Unger S.H. [1964], 'GIT- A heuristic program for testing pairs of directed line graphs for isomorphism', Communications of the ACM, Vol 7, No 1, Jan 1964, pp. 26-35.

Walsh I.L. [1923], 'A closed set of orthogonal functions', Amer. jnl. of mathematics, Vol 55, pp. 5-24.

Ward M.R, Rossol L, and Holland S.W [1979], 'CONSIGHT: A practical vision based robot guidance system', 9th int. symposium on industrial robots, pp. 195-211

Wong V.S. [1979], 'Computational structures for extracting edge features from digital images for real time control applications', PhD thesis, California Institute of Technology.

Yachida M. and Tsuji S. [1977], 'A versatile machine vision system for complex industrial parts', IEEE trans. on computers, Vol C-26, No 9, Sept 1977, pp. 882-894.

Zimmerman N.J, Scheerboom P.L, Steenvoorden G.K, Groen F.C.A [1982], 'Automatic visual inspection system for hybrid circuits', IEEE conf. on industrial applications of machine vision, pp. 55-61.

Appendix 1

Probability of a Random match between two features

This appendix is concerned with obtaining an approximate expression for the likelihood of a match between two local features picked at random. This calculation gives the basic result that may be used for other computations. This appendix also aims to demonstrate that the vocabulary of the feature descriptor is very large.

Due to the complexity involved in obtaining an exact expression,† it is sufficient for our purpose to make a few simplifying approximations in order to establish the scale of the probability figure. I first look at the probability of an exact match occurring at random. Since I am considering <u>random</u> events, I can assume that all (possible) rep-point patterns are equally probable.

Consider Fig. A1-1. The number of rep-point positions possible will be proportional to the area of the local neighbourhood. Thus, there are $C\pi R^2$ positions in which the first peripheral rep-point can be chosen, where C compensates for quantization and is approximately equal to 1. However, since each rep-point has underlying structure, rep-points

† I would like to thank the Napier College statistics group for time spent on this problem.

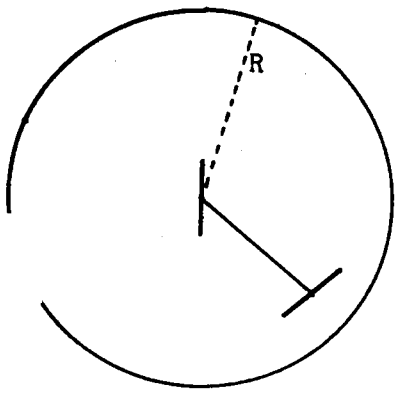


Fig. A1-1 A feature with two rep-points

will occupy a certain number of pixel positions which other rep-points cannot occupy. Assuming that the average number of pixel positions occupied by a rep-point is given by $C\pi r^2$, the first peripheral rep-point can therefore be chosen in only $C\pi(R^2-r^2)$ positions. Since the peripheral rep-point can have D distinct orientations relative to the central rep-point, the total number of possible patterns with two rep-points is given by $C\pi(R^2-r^2)D$. From a similar line of reasoning, the second peripheral rep-point can be chosen in $C\pi(R^2-2r^2)D$ ways, and so there will be

$$C^2\pi^2(R^2-r^2)(R^2-2r^2)D^2$$

patterns with 3 rep-points. (Note that this progression does not strictly hold when the rep-point density within the local neighbourhood increases). It will be noticed that the total number of possible rep-point patterns (and therefore the vocabulary of the feature descriptor) is very large, and increases rapidly when the radius of the local neighbourhood is increased. The probability of a random match between rep-point patterns that contain three rep-points (count-

ing the central rep-point) is then given by

$$\frac{1}{C^2\pi^2(R^2-r^2)(R^2-2r^2)D^2}$$

For an inexact match (i.e. using the matching criterion in section {4.2}), where ρ is the positional variation allowed and d is the orientation variation allowed, the number of rep-points that will match a given rep-point = $C\pi\rho^2 d$

Therefore the random match probability for patterns with 2 rep-points is given by

$$\frac{\rho^2}{(R^2-r^2)}\frac{d}{D}$$

Probability for patterns with 3 rep-points=

$$\frac{\rho^4}{(R^2-r^2)(R^2-2r^2)}\frac{d^2}{D^2}$$

However, the probability of matching two arbitrary neighbourhoods N1 and N2 will be even smaller, as N1 and N2 will not in general have the same number of rep-points. Therefore the random match probability will be further reduced by the distribution of local neighbourhood rep-point numbers. (Section {5.3.3} gives empirical data for variation of rep-point numbers).

For the parameters that I use, the following approximate values hold:

$C\pi R^2=256$, $C\pi r^2<10$, $C\pi\rho^2=14$, $D=256$, and $d=12$

Then for 2 rep-points, the probability = $2.6*10^{-3}$

For 3 = $7.4*10^{-6}$

For 4 = $21.0*10^{-9}$

In fact the unique neighbourhoods chosen by my program often contain neighbourhoods with up to 10 rep-points (including the central rep-

point).

If the radius of the neighbourhoods is doubled, the values above change to:

For 2 $= 0.6*10^{-3}$

For 3 $= 0.4*10^{-6}$

For 4 $= 0.3*10^{-9}$

This result demonstrates that the match probability for two neighbourhoods selected at <u>random</u> is small. It is important to note that this does not necessarily apply to neighbourhoods that are generated by a real scene because (of course) they are not random. Therefore, it is fair to assume that any match that is obtained is due to the original pattern generating mechanism (i.e. the object + imaging conditions) rather than due to a random event. In other words, when a match is obtained between feature f1 and feature f2, the system concludes that this was due to <u>similarity</u> in the object structure that gave rise to the two features. It must be pointed out that this is not the same as saying that f1 and f2 were due to the <u>same object</u>. (i.e. the feature matching algorithm does not <u>recognize</u> features, but computes similarities). It is the task of the learning stage to use such similarity measures to produce an overall recognition capability.

Support for this computation on random match probability comes from the Gaussian noise tests in section {5.2.2.2}.

Appendix 2

User Interface to the Software

The purpose of this appendix is to briefly indicate the form of the software as at present. Firstly, it should be said that the software is not in an industrially usable form, and is mainly geared towards program development. A large amount of the code is devoted to debugging and display of program execution.

There are 6 main processing programs:

WALSH.FTN	- Walsh transform based edge detection
REP.FTN	- Finds rep-points in the edge image
NABOURS.FTN	- Finds neighbourhoods, normalizes them, and forms object models
COMPARE.FTN	- The main learning routine
SORT.FTN	- Sorts the unique features to form the recognition data structure
RECOGNIZE.FTN	- The recognition routine

Communication between these programs is via disc files. This is for ease of program development. The time required to read and write from disc is not included in the processing times reported in section {5.4} because these overheads would not exist in a proper industrial imple-

mentation.

As mentioned earlier, all of these programs contain a large amount of debug code, which is selected (or de-selected) at compile time. The debug code produces a dynamic display of program execution. All data files carry a status area that allows the progress of the data through the different processing stages to be monitored. In addition to the displays provided by the debug code, a number of specialized display programs are also available. They operate directly on the data files and process them for viewing either on a VDU, line printer, or graphics terminal. The most commonly used program (called EDGESHOW.FTN) is able to display grey scale images, edge images and rep-point images in a variety of formats.

In normal use, the interface between the different programs is handled by a few command files. They construct standard data file names and call the processing routines in the correct sequence to allow the system to carry out the processing with a minimum of user effort. The command files also allow the system tests to be run automatically by allowing processing parameters (such as thresholds) to be varied, and transferred to the processing routines when they are called.

However, due to the overheads created by shuffling data between disc and main memory, the total time taken to teach 3 different objects from 15 instances (for example) is about 50% more than the time taken for processing alone. In fact when the system is busy, it can take up to an hour to teach 3 objects. It will be appreciated that under these conditions the total time taken to test the system over a

variety of imaging and operating conditions is very large. (I used 500 images to obtain the required variations of objects and imaging conditions). This explains the earlier comments on the problems of testing the system exhaustively; The number of combinations of different thresholds, different imaging conditions, and different objects, is very large indeed.

The following figures (Fig. A2-1 to Fig. A2-3) give an idea of the programmer interface when using the vision system.

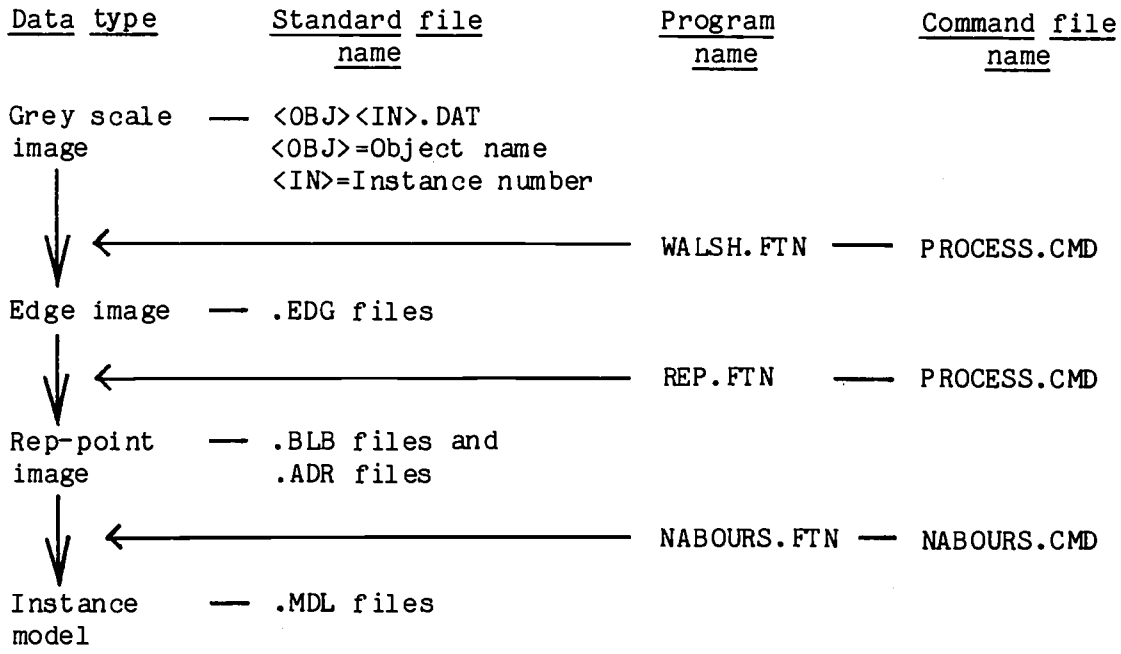


Fig. A2-1 The Pre-Processor

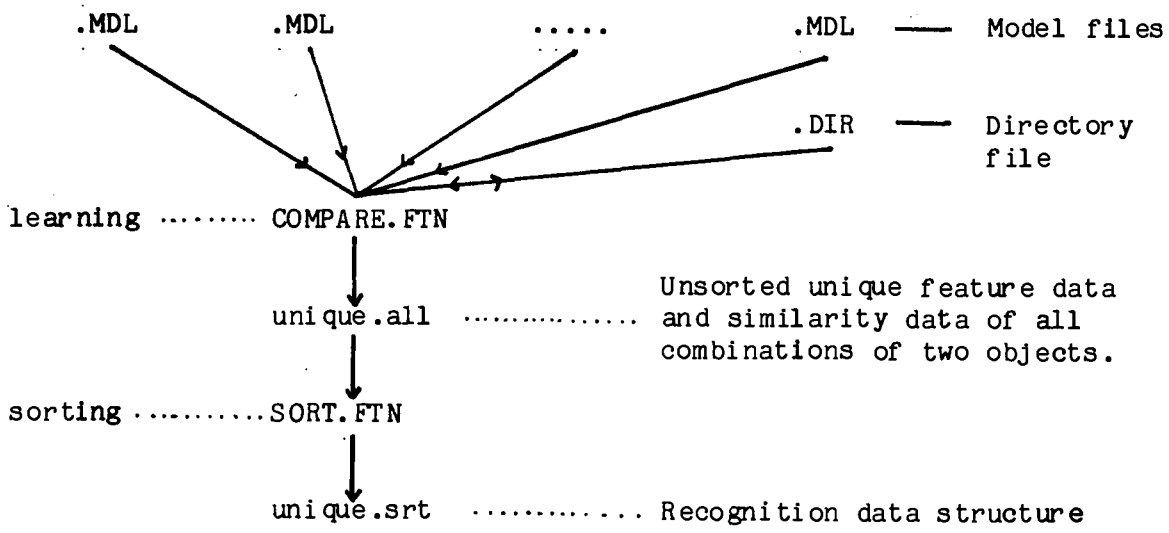


Fig. A2-2 The Learning Stage

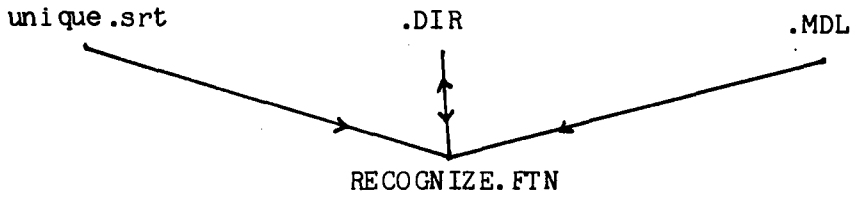


Fig. A2-3 The recognition stage

The command sequence for a typical learning session would be as follows:

```
@PROCESS cutter 1 5     ;Find rep-points in first 5 instances
@PROCESS tooth  1 5     ;of cutter, tooth, and gear.
@PROCESS gear   1 5

@NEWDIR                 ;Create new directory for objects

@NABOURS cutter 1 5 C   ;Form models for each instance of each
@NABOURS tooth  1 5 T   ;object and name the three objects
@NABOURS gear   1 5 G   ;C,T,G, respectively
```

```
@COMPARE                ;Extended learning routine.

@SORT                   ;Sort unique features and create
                        ;recognition data structure
```

Several other command files for activating display programs, and for task building the system, are also available.

Appendix 3

Publications

As required by the university regulations, the following is a list of papers and reports in which parts of this work have been reported.

Athukorala A.S., Low level vision, Dept. of Artificial Intelligence. University of Edinburgh. Working paper No. 66. April 1980

Athukorala A.S., A strategy for recognizing complex objects with operational flexibility, Submitted to IEEE Trans. on Pattern Analysis and Machine Intelligence, May 1984. Also as DAI Research paper No:225, Dept. of Artificial Intelligence, Edinburgh University, June 1984.

Athukorala A.S. [1985], A strategy for recognizing complex objects, Proc. of the 2nd Int. tech. symp. on optical and electro-optical applied science and engineering, SPIE, ANRT, Cannes, France, 2-6 December 1985.

Athukorala A.S. [1985], An essay on the aspects of the human visual system that influenced my thesis work, DAI working paper, Dept. of Artificial Intelligence, Univ. of Edinburgh, UK.

Other papers are also planned.